REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

3

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
August 1989

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Investigation of the Adaptive Discrete Cosine Transform Technique for Still Picture Data Compression

5. FUNDING NUMBERS
C-DCA 100-87-C-0078

TA-88-7

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Delta Information Systems, Inc.
Horsham Business Center, Bldg.3
300 Webb Road, Horsham, PA 19044

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Communications System
Office of Technology & Standards
Washington, DC 20305-2010

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
TIB 89-6

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this study was to investigate the effectiveness of the Adaptive Discrete Cosine Transform Technique when applied to the transmission of gray scale imagery via Group 4 facsimile. The Adaptive Discrete Cosine Transform is a gray scale and/or color coding technique showing the promise of large compression radtios and good picture quality for the coding and transmission of pictorial data. No comprehensive study analyzing the Adaptive Discrete Cosine Transform, as applied to Group 4 facsimile systems, under carefully controlled conditions, has been performed prior to this investigation.

14. SUBJECT TERMS
Group 4 facsimile
Gray Scale
Adaptive Discrete Cosine Transform

15. NUMBER OF PAGES
124

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filing in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements.*

Block 1. Agency Use Only *(Leave blank)*

Block 2. Report Date. Full publication date including day, month, and year, if available (e g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as. Prepared in cooperation with..., Trans. of..., To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA -	Leave blank.
NTIS -	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only).*

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

NCB TIB 89-6

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN
89-6

INVESTIGATION OF THE ADAPTIVE DISCRETE COSINE TRANSFORM TECHNIQUE FOR STILL PICTURE DATA COMPRESSION

AUGUST 1989

INVESTIGATION OF THE
ADAPTIVE DISCRETE COSINE TRANSFORM TECHNIQUE
FOR GROUP 4 FACSIMILE

AUGUST, 1989

SUBMITTED TO:

NATIONAL COMMUNICATIONS AGENCY

Office of Technology and Standards

WASHINGTON, D.C. 20305

Contracting Agency:

DEFENSE COMMUNICATIONS AGENCY

Contract Number - DCA100-87-C-0078

Task Number - 88-7

DELTA INFORMATION SYSTEMS, INC
Horsham Business Center, Bldg. 3
300 Welsh Road
Horsham, PA 19044
TEL: (215) 657-5270 FAX: (215) 657-5273

NCS TECHNICAL INFORMATION BULLETIN 89-6

INVESTIGATION OF THE ADAPTIVE DISCRETE COSINE TRANSFORM TECHNIQUE FOR STILL PICTURE DATA COMPRESSION

AUGUST 1989

PROJECT OFFICER	APPROVED FOR PUBLICATION
	Dennis Bodson
GARY REKSTAD Electronics Engineer Office of Technology and Standards	DENNIS BODSON Assistant Manager Office of Technology and Standards

FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program. Under this program, the NCS, with the assistance of the Federal Telecommunication Standards Committee identifies, develops, and coordinates proposed Federal Standards which either contribute to the interoperability of functionally similar Federal telecommunication systems or telecommunication systems. In developing and coordinating these standards, a considerable amount of effort is expended in initiating and pursuing joint standards development efforts with appropriate technical committees of the International Organization for Standardization, and the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union. This Technical Information Bulletin presents an overview of an effort which is contributing to the development of compatible Federal, national, and international standards in the area of Video Teleconferencing. It has been prepared to inform interested Federal activities of the progress of these efforts. Any comments, inputs or statements of requirements which could assist in the advancement of this work are welcome and should be addressed to:

Office of the Manager
National Communications System
ATTN: NCS-TS
Washington, DC 20305-2010
(202) 692-2124

TABLE OF CONTENTS

TECHNICAL INFORMATION BULLETIN 89-6
INVESTIGATION OF THE ADAPTIVE DISCRETE COSINE TRANSFORM TECHNIQUE FOR STILL PICTURE DATA COMPRESSION

--------------- APPENDIX A ---------------

1.0 Introduction

This document summarizes work performed by Delta Information Systems, Inc., (DIS) for the National Communications System, Office of Technology and Standards. This office is responsible for the management of the Federal Telecommunications Standards Program, which develops telecommunications standards, whose use is mandatory for all Federal departments agencies. The purpose of this study, performed under task order number 88-7 of contract number DCA100-87-C-0078, was to investigate the effectiveness of the Adaptive Discrete Cosine Transform Technique when applied to the transmission of gray scale imagery via Group 4 facsimile.

The Adaptive Discrete Cosine Transform is a gray scale and/or color coding technique showing the promise of large compression ratios and good picture quality for the coding and transmission of pictorial data. No comprehensive study analyzing the Adaptive Discrete Cosine Transform, as applied to Group 4 facsimile systems, under carefully controlled conditions, has been performed prior to this investigation. (KCR)

1.1 Background

At the present time, CCITT Recommendations for Group 4 permit the transmission of only black-white imagery. Consequently, any input page containing gray scale information will be severely distorted by basic Group 4 machines. As a result of much increased commercial interest from, in particular, major computer and telecommunications companies, there has been intense effort in the international standards bodies to select a photographic image compression technique for future image storage and communications applications. The focal point of this activity has been the Joint Photographic Experts Group (JPEG) of ISO/IEC and CCITT.

The JPEG was formed at the end of 1986 under the umbrella of the ISO working group (now ISO/IEC/JTC1/SC2/WG8 - Coded Representation of Picture and Audio Information). It brings together ISO picture coding knowledge with CCITT telecommunications service expertise (from the New Image Communications (NIC) group of CCITT Study Group VIII). Its aim was to select and develop a compression/decompression technique for natural color and gray scale images. The technique will form the basis for both an ISO standard and a CCITT recommendation.

A specification for a compression technique was formulated for a particular range of services and applications including photographic videotex, still picture transmission, document photographic coding and image databases. To support such a range of applications the technique should be adaptable to a wide range of image resolutions and to varying image quality. It should also be capable of providing progressive image build-up (multi-stage with improving quality) or sequential image transmission.

A JPEG meeting was held in Copenhagen in January, 1988, to pick an algorithm from the following candidates: the IBM Adaptive Differential PCM algorithm (ABAC), the European Esprit Discrete Cosine Transform (ADCT) and the NTT Block Separated

Progressive Coding algorithm (BSPC). In subjective testing the ADCT technique achieved considerably higher quality results than the other two techniques. The ADCT technique was therefore selected as the basis for the future standard.

This report is composed of five sections and an appendix. Section 1.0 provides a brief synopsis of the study objectives and outlines its results and conclusions. Section 2.0 describes the Discrete Cosine Transform, discusses coding methods and various transmission modes, and concludes with a brief overview of the simulation system used in this study and a comparison of it with the JPEG system. Section 3.0 describes the simulation system in more detail. Section 4.0 presents image and simulation selection criteria and simulation results. Section 5.0 gives conclusions and recommendations for future investigation. Appendix A discusses factors governing image fidelity and data compression.

1.2 Synopsis

The investigation was conducted in four major, and to some extent overlapping, phases. The first phase consisted of a study of JPEG documentation, and continued as more was released. Simulation software was developed in the second phase with emphasis on the Group 4 facsimile environment and with the aim of supplementing data emerging from the ongoing JPEG investigations. The third phase was devoted to simulating the transmission and reception of JPEG images as well as the Standard Gray Scale Images developed for the NCS in a previous study.[1] The fourth phase consisted of evaluating the results, writing the final report and preparing all deliverable items.

The simulated system is built around the Discrete Cosine Transform (DCT). The starting image is divided into blocks, 8 pixels on a side. Each block is transformed, and the resulting 64 spatial frequency components, conventionally called coefficients, are linearly quantized according to a "visibility matrix," which defines the quantum step size for each coefficient. The resulting quantum numbers are then losslessly encoded and transmitted. The receiver decodes the quantum numbers, constructs the quantized coefficients and performs the inverse DCT to recover an approximation of the starting image.

JPEG has proposed a variety of systems with the DCT as their centerpieces. The most basic is called sequential transmission, and consists solely of the DCT compression method just described. Another is called progressive build-up, and it has two major variations: (1) hierarchical progression, consisting of transmitting first a low-quality image with very high compression and then a sequence of image refinements, which permit the receiver to build improving approximations of the original image; and (2) bit slicing, in which successive corrections to the coefficient quantum numbers, as contrasted to image refinements, are transmitted. The second method, which has been extensively investigated by JPEG, has the advantage that the inverse transform is performed by the receiver only. Hierarchical progression requires that the transmitter and the receiver both perform the inverse DCT, and both must do so for each refinement.

A further method is lossless transmission, i.e. transmission with zero distortion. JPEG is currently leaning toward a form of DPCM instead of the DCT because of the lack of a standard DCT algorithm, which would be required to have sufficient precision to guarantee zero distortion. Another approach being considered is progressive build-up with a final correction transmitted by some form of DPCM. This approach has the drawback of requiring that the transmitter and the receiver use the same inverse DCT algorithm, so the transmitter "knows" exactly what image the receiver is reconstructing. Without this requirement, the final correction image might not exactly correct the approximate image reconstructed by the receiver. Because the Group 4 requirement for lossless transmission has not been defined, the investigation concentrated on other aspects of the JPEG study.

Simulations of sequential transmissions yielded compressed data bit rates ranging from 0.7 to 1.5 bits per pixel, depending on the image, with very good image quality. Hierarchical progressions with sub-sampling and interpolation gave a few percent worse compression than sequential transmissions. A bit slicing progression, described in JPEG literature, but not simulated in this study, yields a few percent better compression than the sequential method at the expense of much greater system complexity.

2.0 Investigation

The investigation began with a study of JPEG document ISO/JTC1/SC2/WG8 N800, "Coded Representation of Picture and Audio Information," May, 1988, henceforth to be called Doc. N800. This document describes a system to perform hierarchical and sequential image transmission. The major features of this system are:

- The Discrete Cosine Transform (DCT);
- Uniform coefficient quantization controlled by a "visibility matrix" (since renamed "quantization matrix") defining the quantum step size for each coefficient;
- A quantization scale factor (referred to as FACTOR in JPEG literature), which scales the visibility matrix; the greater this factor, the greater the data compression, but also the greater the distortion in the reconstructed image;
- Lossless transmission of the coefficient quantum numbers (quantum step numbers) for each image block in order of increasing spatial frequency;
- Organizing the quantum number data to make efficient use of Huffman coding, transmitting Huffman coding tables optimized for the image, and transmitting the actual Huffman-coded data;
- Decoding the quantum numbers and "dequantizing" the coefficients (multiplying the quantum numbers by the coefficient step sizes);
- Performing the inverse DCT and constructing an approximation of the original image;
- Compression by filtering and sub-sampling, expansion by interpolation, and image addition and subtraction, all to support hierarchical progression.

As the study progressed, additional JPEG documents were released. These documents describe:

- Progressive build-up of the coefficient quantum numbers (bit slicing),[2][3] as contrasted to hierarchical progression;

- The theory of and flow diagrams for the Q Coder,[4][5] a patented IBM binary arithmetic coder which optimizes the code "on the fly."

This investigation led to the development of what is henceforth called the Simulation System (SS or DIS). In essence, the Simulation System implements the Doc. N800 system, except that it uses the Q Coder instead of Huffman coding, and it does not support color.

The Q Coder was used because published JPEG results were obtained with the Q Coder, not Huffman coding. Color is not addressed in this study because it has not been defined for Group 4 facsimile.

The Simulation System design criteria and a comparison of this system and the JPEG systems are presented in detail in Section 2.9.

The essential features of the various proposals emerging from the JPEG investigation are described below.

2.1 The Discrete Cosine Transform

All the JPEG proposals are built around the Discrete Cosine Transform.

This transform and its inverse are formally defined[6] by the first pair of equations in Figure 2.1, where f(m,n) (m = row, n = column) are the pixel values in an N by N block, F(u,v) (u, v = horizontal and vertical spatial frequency indices) are the horizontal and vertical spatial frequency components ("coefficients"), and c(u,v) is defined to have the value 1/2 for u = v = 0, the square root of 1/2 for u = 0 or v = 0, but not both, and 1 for neither u nor v equal to 0.

Discrete Cosine Transform (Formal Definition)

$$F(u,v) = [4c(u,v)/N^2] \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m,n) \cos[(2m+1)u\pi/2N] \cos[(2n+1)v\pi/2N]$$

Inverse Discrete Cosine Transform (Formal Definition)

$$f(m,n) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u,v)\, F(u,v) \cos[(2m+1)u\pi/2N] \cos[(2n+1)v\pi/2N]$$

Discrete Cosine Transform (DIS and JPEG Implementations)

$$F(u,v) = [2c(u,v)/N] \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m,n) \cos[(2m+1)u\pi/2N] \cos[(2n+1)v\pi/2N]$$

Inverse Discrete Cosine Transform (DIS and JPEG Implementations)

$$f(m,n) = 2/N \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u,v)\, F(u,v) \cos[(2m+1)u\pi/2N] \cos[(2n+1)v\pi/2N]$$

Figure 2.1 Discrete Cosine Transforms (See text for symbol definitions)

The SS and JPEG implementations both express the forward and inverse DCT as shown by the second pair of equations in Figure 2.1. These equations give transform coefficients which are a factor of N/2 times those obtained by the formal definition. The factor $4/N^2$ in the formal definition may, in fact, be distributed between the forward and inverse transform expressions in any manner desired. The actual choice is based on such practical considerations as the hardware or firmware transform implementations.

Of all the various transforms employed for image compression, the DCT is one of the best, for two important reasons. The first is that it has low susceptibility to the blocking artifact.[7] The second is that the DCT comes closest to the Karhunen-Loeve (K-L) transform[8] in energy compaction,[9] that is, the packing of most of the energy of a block of data into a few uncorrelated coefficients. The K-L transform is picture-dependent, requiring intensive computation and the transmission of the transform basis functions for each frame. The DCT is a fixed transform, known to both transmitter and receiver, and performs almost as well as the K-L transform.

2.2 Coefficient Quantization

To achieve data compression, the DCT coefficients must be quantized. The method specified in Doc. N800 and used by DIS is to quantize each coefficient uniformly, under the control of a scaled "visibility" matrix specifying the quantum step size for each coefficient in a block. The unscaled visibility matrix is a constant parameter, designed by JPEG by subjective image quality evaluation. The matrix used in this study is the default luminance matrix specified by JPEG. The system can be adapted to a wide range of image resolution and varying image quality by substituting custom visibility matrices.

For any given transmission, the compression vs. quality trade-off is determined by a quantization scale factor. In the Doc. N800 algorithm, each element of the constant visibility matrix is multiplied by this factor and then divided by 50, all in integer arithmetic. For example, two unscaled matrix elements of values 8 and 16 and two scale factors of values 50 and 55 would give:

(8 X 50)/50 = 8; (8 X 55)/50 = 8;

(16 X 50)/50 = 16; (16 X 55)/50 = 17.

Thus, small changes in the scale factor have no effect on small visibility matrix elements, but do affect larger elements.

2.3 Coefficient Ranking

After the coefficients have been quantized, the quantum numbers must be arranged in some suitable order for encoding and transmission. This process is called ranking, and good ranking enhances compression by placing most of the zero-valued quantum numbers last, where they can be ignored. An end-of-block value or flag tells where the last non-zero quantum number is.

The JPEG and Simulation Systems rank the quantum numbers in "zigzag" order, i.e. in order of increasing spatial frequency.

This principle is illustrated for a 4 by 4 block; the actual blocks are 8 by 8.

Natural Order

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Zigzag Order

1 2 5 9 6 3 4 7 10 13 14 11 8 12 15 16

This method takes advantage of the fact that high-frequency quantum numbers usually have lower magnitudes than low-frequency ones, and are therefore more apt to be zero. While, for any given image, there may be a better ranking, the zigzag method is simple and image-independent.

2.4 Coding Methods

Doc. N800 specifies Huffman coding of the coefficient quantum numbers. More recent JPEG literature discuses the Q Coder, a binary arithmetic coder patented by IBM.

2.4.1 Huffman Coding

Doc. N800 specifies the transmission of the DC (zero-frequency) coefficient quantum numbers by Huffman-coded DPCM, that is, the difference between the DC quantum numbers of the current and previous blocks. Zero-valued AC quantum numbers are not transmitted directly. Instead, each non-zero AC quantum number is transmitted in two data items. The first is a Huffman-coded compound item specifying (a) the number of consecutive zero-valued quantum numbers preceding the non-zero quantum number, and (b) the number of significant bits of the non-zero quantum number. The second data item is a straight PCM code for the significant bits themselves, which are usually few in number.

2.4.2 The Q Coder

The Q Coder is a new binary arithmetic coder invented and patented by IBM. In head-to-head comparisons of the Q Coder with adaptive Huffman coding (coding tables optimized to the data) in DCT-based transmissions, the Q Coder has consistently achieved approximately 10 percent better compression.[10]

Rather than delving into the inner workings of the Q Coder, which are thoroughly described in JPEG literature already cited, this report presents a simple description of what the Q Coder does.

Consider, for example, the case of a binary image (black and white, no intermediate gray levels). Uncompressed data for such an image require 1 bit per pixel. As a binary image is scanned, runs of two or more pixels of the same value are very frequently encountered, and long runs of the same value are common. The Q Coder continuously keeps track of the local probabilities (frequencies averaged over many pixels, but not the whole image) of the two symbols ("black" and "white"). The one currently occurring the more frequently is called the more probable symbol (MPS), and the other is called the less probable symbol (LPS). The more probable the MPS, the lower the bit rate for it, and the higher the bit rate for the LPS. Since the MPS occurs more frequently, the average bit rate for both symbols decreases as the probability of the MPS increases.

The Q Coder adapts to local statistics. If, for example, a long run of "blacks" is followed by a long run of "whites," high compression is achieved during most of both runs, but the bit rate per symbol increases considerably during the transition between them. Because of these transitions, random binary data in general give no compression and sometimes give expansion (more than one bit per original bit).

What was just described is an example of a single-context model. Most systems employing the Q Coder require a multiple-context model. For example, to Q-code multiple-way decisions, the decisions must be mapped into binary trees, the Q Coder encoding each binary decision in a given tree. Each decision in the tree may have different statistics from those of other decisions in the same tree, and should therefore be considered in a separate context to take advantage of these different statistics. Another example of multiple contexts is a set of binary decisions, not necessarily comprising one tree, having significantly different statistics.

The Q Coder can track separately and simultaneously any reasonable number of contexts, limited only by available memory, keeping local statistics for each. The contexts share a common probability table; hence, each context requires storage only for its MPS value and a pointer into the probability table. Each context exhibits high compression when its MPS is much more frequent than its LPS. Its bit rate increases only when the frequency of its LPS increases.

Compression can be further enhanced by taking advantage of correlation among various binary decisions. This is called conditioning, and is employed widely in the JPEG Q Coder models. For example, binary image compression could be improved by the use of two contexts: (1) current pixel is preceded by a white pixel, and (2) current pixel is preceded by a black pixel. The current pixel would most of the time be white in context 1 and black in context 2.

2.5 Baseline System

The latest JPEG standards proposals specify a required baseline system consisting of:

- o Sequential transmission by DCT with some loss of fidelity permitted;
- o Lossless Huffman coding of the coefficient quantum numbers with default coding tables;
- o Resynchronization capability.

Extended system options include:

- o Lossless (distortionless) transmission;

- o Progressive build-up by image hierarchy or quantum number bit- slicing;

- o Optimizing and transmitting picture-dependent Huffman coding tables;

- o Arithmetic, instead of Huffman, coding.

For Group 4 facsimile, some of the baseline system requirements may be dropped. Resynchronization would not be required if error-free transmission of the compressed data is in fact guaranteed and always occurs. The JPEG system includes color; color has yet to be defined for Group 4 facsimile.

2.6 Sequential Transmission

Sequential transmission, the simplest transmission mode investigated, consists of dividing the starting image into N by N blocks (8 by 8 in the proposed standard), and transforming, quantizing, encoding and transmitting each block as it is extracted from the image. This method achieves good compression while requiring image storage by the transmitter and receiver of only N rows of pixels, not the whole image.

2.7 Lossless Transmission

Since the Group 4 facsimile requirement for lossless transmission has not been defined, it is discussed only briefly in this report. Lossless transmission can be achieved in one pass or more than one. In the one-pass method the entire image is transmitted once without any distortion. In the multiple-pass method, the image is first transmitted with some distortion in one or more passes, and then a correction "image" is transmitted and added to the approximate image to yield an exact copy of the original image.

2.7.1 One-Pass Transmission

JPEG recommends against the employment of the DCT in one-pass lossless transmission for two reasons: (1) The coefficients would have to be represented in sufficiently high precision to ensure an exact reconstruction of the image by the inverse DCT, and (2) fast forward and inverse DCT algorithms (hardware or firmware implementations) would have to be standardized. Instead, JPEG recommends lossless differential pulse-code modulation (DPCM) consisting of a simple prediction method, quantized error values, and the transmission of corrections to eliminate the quantization noise.[11]

2.7.2 Multiple-Pass Transmission

Doc. N800 specifies a hierarchical progression of DCT-based transmissions (described below) to achieve increasingly accurate approximations of the original image. A correction "image" is then losslessly transmitted. This method allows a user on the receiving end to view the improving image as it is refined in successive transmissions. It has, however, the important drawback of requiring that both the transmitter and receiver perform the inverse DCT and, moreover, that both employ exactly the same algorithm with the same data precision. Without this requirement, the transmitter could not be guaranteed to transmit the exact correction to the approximate image at the receiver. Because DCT algorithms have not yet been standardized,[12] JPEG now recommends against a DCT-based hierarchical progression for the lossless case.[10]

2.8 Progressive Build-up

"Progressive build-up" refers to the transmission of a low-quality image with very high compression, followed by refinements that successively improve the image quality. Progressive build-up can be performed in either the image or the transform domain.

2.8.1 Hierarchical Progression

Hierarchical progression is a form of progressive build-up in the image domain. Doc. N800 specifies the following operations:

(1) The transmitter filters and sub-samples the original image, I_1, to produce image I_4, which contains half as many pixels per row and half as many rows as the original image, i.e., one-fourth the total number of pixels;

(2) The transmitter similarly sub-samples image I_4 to produce I_{16}, having one-sixteenth the number of pixels as the original image;

(3) The transmitter transmits (performs the DCT, quantizes, encodes and transmits the quantum numbers for each image block) Image I_{16} at a fairly high bit rate per transmitted pixel. The effective bit rate for the full-sized image is one-sixteenth the transmitted rate because of the sub-sampling, and is therefore

[10] Annex 4 of Doc. 499R of CCITT SGXV Working Party XV/1, "Specialists Group on Coding for Visual Telephony," March 10, 1989.

very low.

(4) Both transmitter and receiver multiply the coefficient quantum numbers by their step sizes, giving the quantized coefficients, and perform the inverse DCT to produce image I_{16}', an approximation of I_{16}. They then expand by interpolation I_{16}' to produce I_4'. Image I_4' is an approximation of the I_4 produced in Step (1). The receiver may expand I_4' and display the resulting full-sized I_1', the first approximation of the original image.

(5) The transmitter transmits a difference image, $I_4 - I_4'$, at a higher effective bit rate than that employed in step (3). Both transmitter and receiver add the <u>received</u> (dequantized, inverse-transformed) version of this difference image to I_4', giving I_4'', a better approximation of I_4.

(6) Both transmitter and receiver expand by interpolation I_4'' to produce I_1'', a full-sized second approximation of the original image, I_1.

(7) The transmitter transmits the difference between I_1 and I_1'' to improve the received image quality. Additional refinements may be transmitted if desired. (It is shown later that such additional refinements considerably degrade data compression.) Doc. N800 specifies an optional lossless transmission of a final refinement to give an exact reproduction of the original image.

The total bit rate for the whole sequence is the sum of the <u>effective</u> bit rates (bits per pixel of the <u>full-sized</u> image) of all the transmissions. Good compression is achieved because: (1) the effective bit rate for sub-sampled images is low, and (2) since the "pixel" values of difference "images" cluster around 0 (because they represent refinements to already good approximations), so do the transform coefficients; hence, there are many zero-valued coefficient quantum numbers, and the magnitudes of non-zero quantum numbers are usually small. Nevertheless, Doc. N800 states that for a given total effective bit rate, hierarchical progression gives final images of <u>lower</u>

quality than does sequential transmission. Conversely, for a given image quality, the effective bit rate is higher for hierarchical than for sequential transmission. Simulations performed in this study produced similar results. Possible causes of this effect are presented in Appendix A.

2.8.2 Bit Slicing

Bit slicing, described in recent JPEG literature,[13] is a form of progressive build-up in the transform domain. This, and any transform-domain progressive build-up method, has the advantage of not requiring the transmitter to perform the inverse DCT. Because bit slicing, in conjunction with the Q Coder, has been found to produce the best compression of various methods tested,[14] it may become the standard for the extended system options.

The transmitter performs the DCT and quantizes the coefficients to a quantization scale factor of 25. This would give very good image quality in a sequential transmission.

The transmitter divides the resulting quantum numbers by 8 and losslessly encodes and transmits the resulting quotients for each block in the entire image. Only quotients through the end-of-block position (last non-zero quotient) are transmitted, since the remaining quotients are known to be zero. The receiver multiplies the (losslessly) transmitted quotients by 8, thus obtaining coarse approximations of the original quantum numbers. This is equivalent to a sequential transmission with a quantization scale factor of 200 (8 times 25). The receiver may perform the inverse DCT to display the first image approximation. Non-zero quotients transmitted in this first pass are referred to as having "history;" the corresponding quantum numbers are known not to be zero, and their signs are known.

In the second pass the transmitter divides the original quantum numbers by 4. For each quotient through the (probably greater) end-of-block position, the transmitter sends to the Q Coder for compression:

- o For each quantum number previously known not to be zero (has "history"), a one-bit refinement to its current approximation;
- o For each quantum number whose previous zero/not-zero status is still unknown ("still zero" in JPEG notation), a zero/not-zero bit, and, if the quantum number is now known to be not-zero, a sign bit.

All newly-found non-zero quantum numbers are added to the list of those having "history."

In a third pass, the transmitter divides the original quantum numbers by 2 and repeats the process described for the second pass. A final pass refines the quantum numbers to their original values.

It is emphasized that this bit-slicing method is based on binary arithmetic coding, e.g., the Q Coder, not Huffman coding. Whether bit slicing would be effective with Huffman coding (of run lengths of zero values, for example) is a matter of conjecture.

In JPEG experiments, bit slicing gave a few percent better compression for a given image quality than the sequential method, which also used the Q Coder. Pennebaker and Mitchell[15] attribute this result to the fact that the least significant bit of a quantum number has noise-like statistics, whereas higher bits have signal-like statistics. In the sequential mode the statistics for all the magnitude bits of a quantum number are lumped together with the result that the "noise" bit interferes with compression of the "signal" bits. In the bit slicing method the statistics of the noisy least significant bit are isolated from those of the others. It is conjectured that bit slicing each quantum number separately in a sequential transmission might similarly improve compression by isolating the "noise" bit statistics.

For bit slicing to compress as well as, let alone better than, sequential transmission, all history (end-of-block, zero/not-zero, sign, currently-known magnitude, etc.) must be meticulously maintained throughout the entire transmission sequence. This requires that the transmitter and receiver store all the quantum numbers for the entire image. Bit slicing therefore requires one to two orders of magnitude more memory than the sequential method.

2.9 The Simulation System

2.9.1 Design Criteria

The most important design criterion was to ensure that the Simulation System match the JPEG system in terms of producing the same coded image for a given starting image, quantization scale factor and transmission mode. In other words, the two systems must be functionally identical with the possible exception of data compression. The only other exceptions are that the Simulation System need not support resynchronization or color since the ADCT approach is being evaluated for the transmission of gray-scale imagery by Group 4 facsimile.

The second criterion was the ability to compare Sumulation System and JPEG data compression for sequential transmissions and to evaluate various progressive build-up methods. Because the published JPEG results were obtained with the Q Coder, the Simulation System also employs the Q Coder instead of Huffman coding.

The third criterion was simplicity owing to budget and time constraints. Bit-slicing was not simulated because of its complexity and memory requirements and because JPEG has thoroughly investigated this approach. The Q Coder model for this study is greatly simplified, even though it would yield somewhat less compression than the JPEG model in any given transmission mode.

The resulting Simulation System was employed to compare various forms of progressive build-up with sequential transmission with the knowledge that all transmission modes would achieve better compression with the more complex Q Coder model.

2.9.2 Comparison of Simulation and JPEG Systems

Some of the methods evaluated by JPEG employ "DC correction" and/or "AC prediction."[16] The DC coefficients are predicted separately[17] and correction values are transmitted. Low frequency AC coefficients are predicted from the DC coefficients of the current and neighboring blocks.[18] These methods are designed to produce more zero-valued quantum numbers and smaller-magnitude non-zero ones. The Simulation System performs no prediction and treats a DC coefficient like any other.

The JPEG Q Coder model conditions the end-of-block decision (Is this non-zero quantum number the last in the block?) on whether the current position in the zigzag ranking is (is not) greater than the known end-of-block position of the previous block. The JPEG model also conditions zero/not-zero, sign and magnitude decisions for a given quantum number on whether or not the zero/not-zero status of the previous quantum number in the same block is known. The study Q Coder model does not include conditioning.

JPEG results reveal that coefficient prediction yields only a few percent better compression than no prediction. On the other hand, the JPEG bit rates for sequential transmissions of four JPEG test images were approximately 0.1 bit per pixel lower than the bit rates. Percentage differences ranged from 9 to 24 with reference to the JPEG results, the greater percentage differences occurring in images for which the rates were low with both systems. Thus, the more sophisticated JPEG Q Coder model yields significantly better compression.

3.0 System Description

3.1 Overview

The ADCT Simulation System is built around a sequential transmission simulator shown in Figure 3.1. Program ENCODE performs the DCT on each 8 by 8 image block, quantizes the coefficients according to a quantization scale factor specified at run time, and saves all the coefficient quantum numbers for the entire image in a file. It then "dequantizes" the coefficients (multiplies the quantum numbers by the quantum step sizes) and performs the inverse DCT to arrive at an approximation of the original image. The original image and its approximation can be compared visually and/or by computing the RMS (root-mean-square) error.

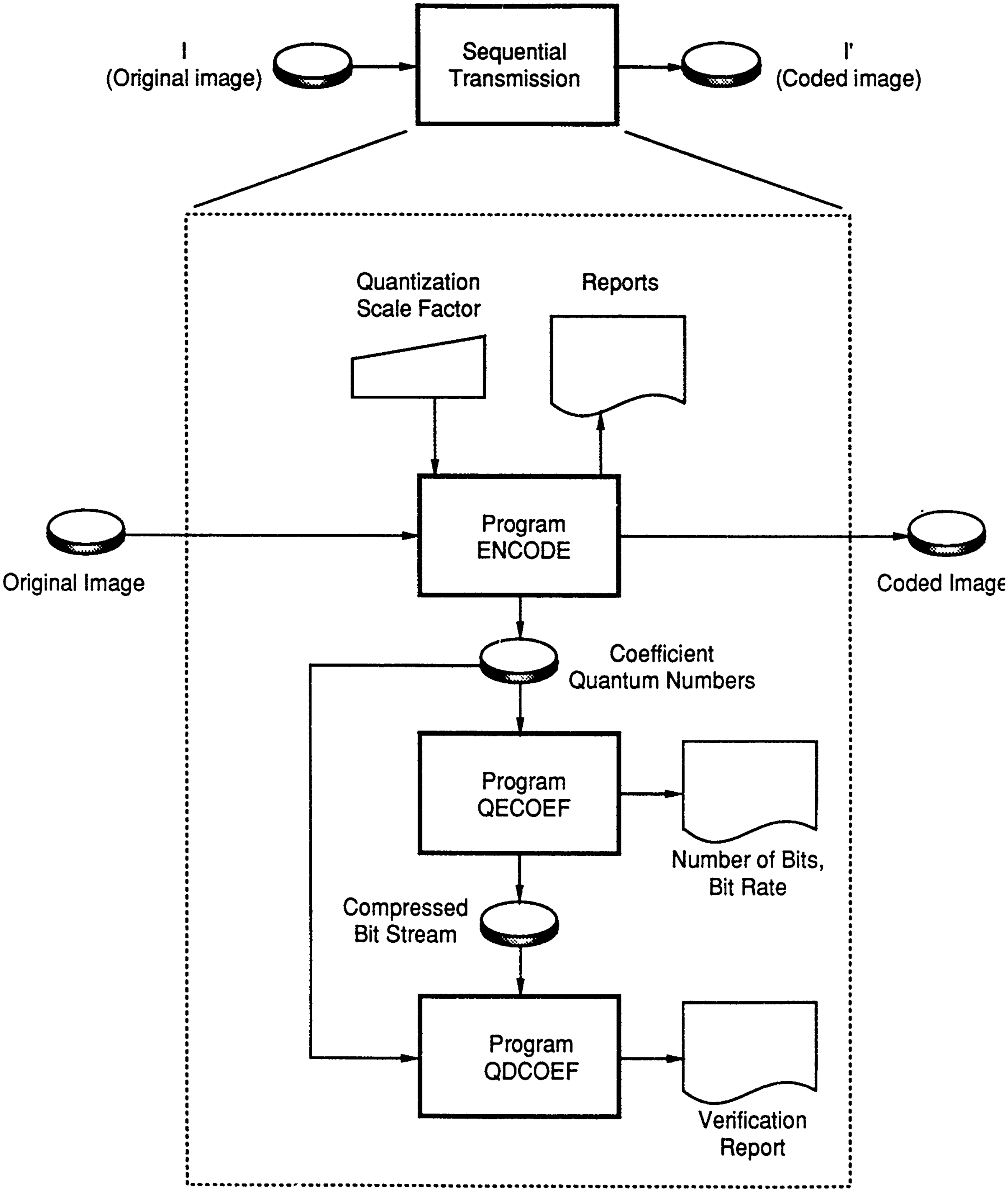


Figure 3.1 Sequential Transmission Subsystem

A separate program, QECOEF, Q-codes the saved coefficient quantum numbers and computes the bit rate. The Q Coder model is a simplified form of that employed by JPEG. Program QDCOEF verifies the integrity of QECOEF by decoding the coefficient quantum numbers from the compressed bit stream and verifying that they match the original quantum numbers.The system also contains programs to:

o Filter and sub-sample an image to half its size in each direction;

o Interpolate a sub-sampled image to twice its size in each direction;

o Produce the pixel-by-pixel difference between two images;

o Produce the pixel-by-pixel sum of two images.

These programs were written to evaluate various forms of progressive build-up. Figure 3.2 shows how all the programs are employed to simulate a 3-pass hierarchical progression.

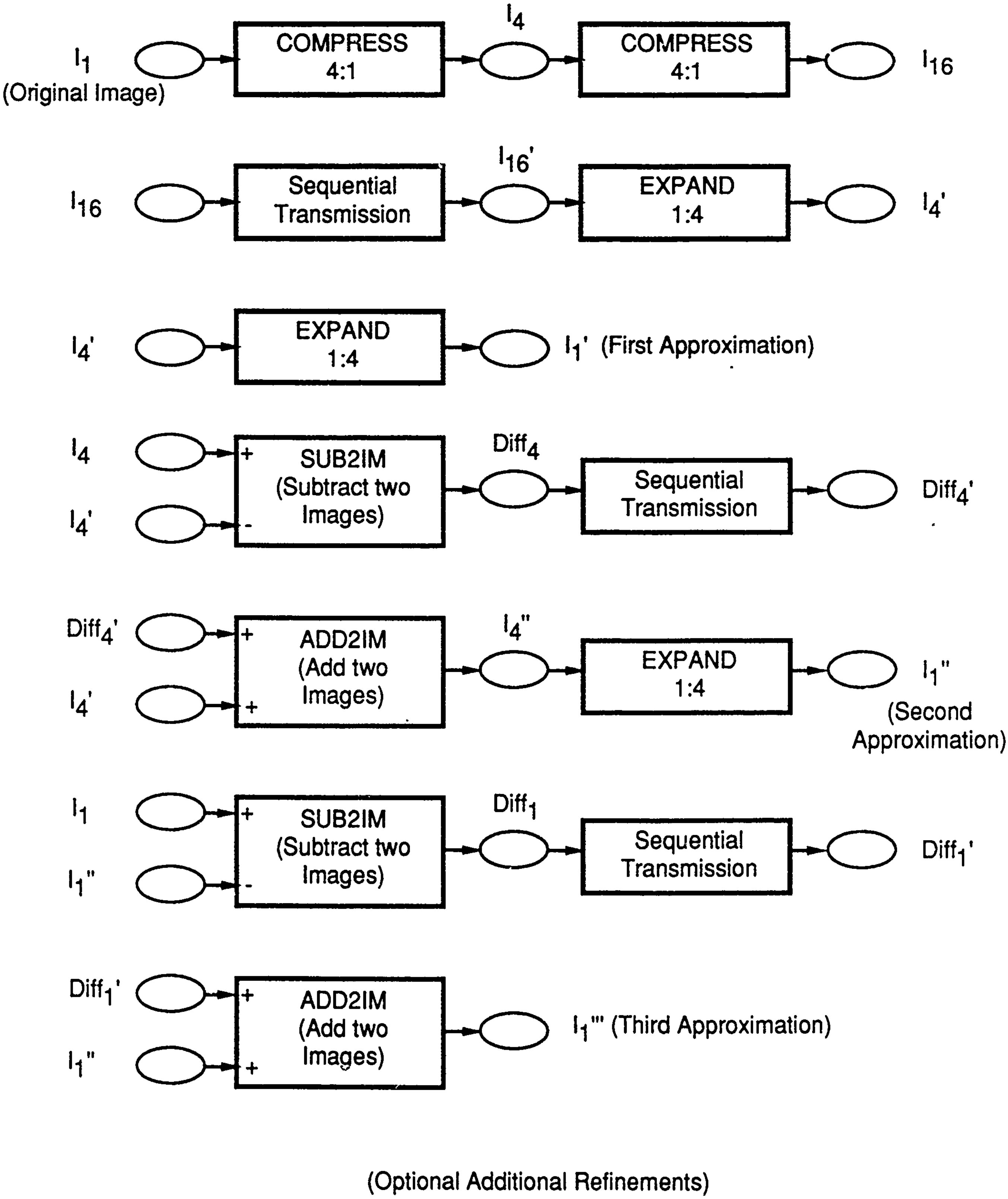


Figure 3.2 Hierarchical Progression (Three Transmission Passes Shown)

3.2 The Baseline System

3.2.1 Program ENCODE

Program ENCODE begins by soliciting the starting image width and height in pixels and the quantization scale factor, S. It next performs the forward DCT transform on each N by N block of the image by the matrix multiplication

$$[F] = [P][f][Q]$$

where [F] is the DCT of the block, [P] is the DCT matrix shown in Figure 3.3, [f] is the block of pixels and [Q] is the transpose of [P], i.e., the rows of [Q] are the columns of [P]. In the current context, N = 8. This matrix multiplication is equivalent to the forward DCT defined in the second pair of equations in Figure 2.1. Each element of the coefficient matrix, [F], is computed in real arithmetic and then rounded to the nearest 16-bit integer.[2]

[2] DCT algorithm with possible slightly less precision. This might account for the very small differences in RMS errors found when DIS and JPEG results were compared for JPEG images.

$$
\begin{array}{c|cccc}
j \backslash i & 0 & 1 & & N\text{-}1 \\
\hline
0 & \sqrt{\frac{1}{N}} & \sqrt{\frac{1}{N}} & \bullet\quad\bullet\quad\bullet & \sqrt{\frac{1}{N}} \\
1 & \sqrt{\frac{1}{N}}\cos\frac{(0.5)\pi}{N} & \sqrt{\frac{1}{N}}\cos\frac{(1.5)\pi}{N} & \bullet\quad\bullet\quad\bullet & \sqrt{\frac{1}{N}}\cos\frac{(N\text{-}1+.5)\pi}{N} \\
2 & \sqrt{\frac{1}{N}}\cos\frac{(0.5)2\pi}{N} & \sqrt{\frac{1}{N}}\cos\frac{(1.5)2\pi}{N} & \bullet\quad\bullet\quad\bullet & \sqrt{\frac{1}{N}}\cos\frac{(N\text{-}1+.5)2\pi}{N} \\
 & \bullet & & \bullet\quad\bullet\quad\bullet & \\
 & \bullet & & \bullet\quad\bullet\quad\bullet & \\
 & \bullet & & \bullet\quad\bullet\quad\bullet & \\
N\text{-}1 & \sqrt{\frac{1}{N}}\cos\frac{(0.5)(N\text{-}1)\pi}{N} & \sqrt{\frac{1}{N}}\cos\frac{(1.5)(N\text{-}1)\pi}{N} & \bullet\quad\bullet\quad\bullet & \sqrt{\frac{1}{N}}\cos\frac{(N\text{-}1+.5)(N\text{-}1)\pi}{N}
\end{array}
$$

Figure 3.3 Discrete Cosine Transform Matrix

Next, ENCODE scales the fixed visibility matrix, [V] to [V'] by multiplying each element of [V] by the quantization scale factor, S, and dividing by 50, all in integer arithmetic. The fixed visibility matrix, [V], is the JPEG default luminance visibility matrix:

$$\begin{matrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{matrix}$$

ENCODE quantizes the coefficients by dividing each element of [F] by the corresponding element of [V'] and rounding the results to the nearest integer. It then rearranges the resulting <u>quantum number</u> (step number) matrix into a vector of 16 elements in zigzag order as described in Section 2.3 The quantum number vectors for all blocks in the image are saved in a file for program QECOEF, described below. Except for data compression, this completes the transmitter simulation.

The receiver is simulated by the same program, instead of a separate one, because a hierarchical progression requires that the transmitter "know" the coded image the receiver constructs. Since the quantum numbers are transmitted losslessly, the bit stream encoded by program QECOEF need not be decoded for functional simulation.

Program ENCODE simulates reception by rearranging the quantum number vectors back into matrix form and then multiplying each element of that matrix by the corresponding elements of the scaled visibility matrix, [V']. (In an actual system, the receiver must, of course, "know" the quantization scale factor, either a-priori or by reception, e.g. one byte.) The resulting quantized coefficients, [F'], are approximations of the original coefficients, [F], with the quantization error of a given coefficient less than or equal to half the step size specified in the scaled visibility matrix, [V'].

Program ENCODE finally performs the inverse DCT on each block of quantized coefficients to yield an approximation of the starting image. The method is the same as for the forward DCT except that matrices [P] and [Q] are interchanged:

$$[f'] = [Q][F'][P]$$

where the primes (') indicate approximations of the actual values because of quantization. This matrix equation is equivalent to the inverse DCT defined in the second pair of equations in Figure 2.1. Were it not for coefficient quantization (and limited data precision), the inverse transform would yield the original image.

3.2.2 Program QECOEF

Program QECOEF encodes the coefficient quantum numbers saved by program ENCODE and writes the compressed bit stream to a file.

The program drives subroutines that simulate the actual Q Coder, which is thoroughly documented in JPEG literature already cited. Only the model used to encode the quantum numbers is described here.

The model employs four contexts for each quantum number in a block: one for the end-of-block decision and the other three for the quantum number value. End-of-block is defined as the position of the last non-zero quantum number in the block vector (the block quantum numbers arranged in zigzag order).

A block of quantum numbers is encoded by the following sequence, repeated until an end-of-block test result is affirmative:

1. Test for end-of-block, first and after encoding each non-zero quantum number, and encode the test result (send a bit to the Q Coder). The first end-of-block test is required in case all quantum numbers in the block are zero. No end-of-block test is required if the last (64th) quantum number is non-zero, since it is known to be last. Therefore, 64 contexts are sufficient for all possible end-of-block tests.

2. If not end-of-block, encode the next quantum number, zero or not, as described below.

Encoding a quantum number involves three contexts: one for the zero/not-zero decision, one for the sign decision if the quantum number is not zero, and one for encoding a non-zero magnitude. (The JPEG model uses six contexts: the three cited above conditioned on the previous quantum number being still zero, and three similar contexts conditioned on the previous quantum number not still zero, where "still zero" means that the zero/not-zero status is not yet known.) The flow diagram is shown in Figure 3.4. In the figure, "Encode 1" and "Encode 0" mean "send the appropriate bit to the Q coder with the proper context."

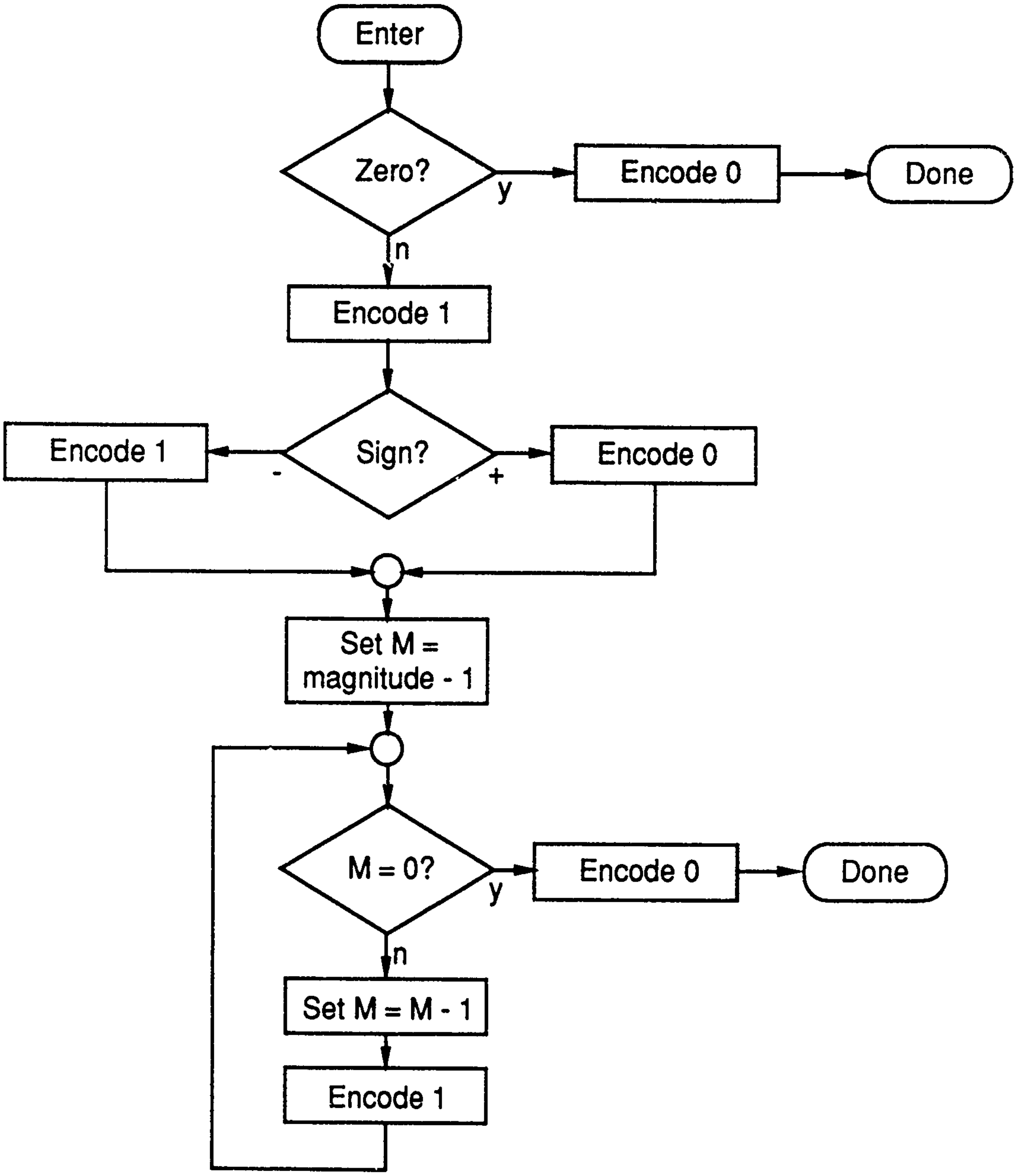


Figure 3.4 Flow Diagram for Encoding One Quantum Number

The flow diagram shows that the magnitude of a non-zero quantum number is encoded by sending to the Q Coder 0 for 1, 10 for 2, 110 for 3, etc.. At first sight, one might think it more efficient to encode the actual magnitude bits, because there would be fewer bits to encode. However, if one examines average entropy, one finds that the method actually employed is more efficient when the quantum numbers are narrowly distributed around 0, which is almost always the case.

3.2.3 Program QDCOEF

Program QDCOEF was written for the sole purpose of verifying the integrity of the Q Coder simulation software. It retrieves compressed data from the bit stream file and decodes all the binary decisions in the order in which QECOEF encodes them, using the same contexts. (In any system employing binary arithmetic coding, the receiver must know what binary decisions are being made when and in what contexts.) For each block of quantum numbers, QDCOEF decodes the quantum numbers from the compressed bit stream and compares them with those saved by Program ENCODE. Any discrepancy would signify a software failure. In initial simulations, program QDCOEF was always executed, and no failure occurred. In later simulations the program was executed only occasionally.

3.3 Progressive Build-up

To simulate various forms of progressive build-up, the following programs were written: COMPRESS to filter and sub-sample an image to half the number of pixels per line and half the number of lines; EXPAND to restore a sub-sampled image to its original size by bi-linear interpolation; SUB2IM to produce the pixel-by-pixel difference between two images; and ADD2IM to produce the pixel-by-pixel sum of two images. The algorithms, summarized below, are identical to those specified in JPEG Doc. N800.

3.3.1 Program COMPRESS

Program COMPRESS solicits the image dimensions, both of which must be even. It then filters every second pixel in every second line and stores the results in the output image. The filtering algorithm consists of computing a weighted average of a pixel to be filtered and its neighbors in the starting image. The weights are shown in the diagram below, where the center weight (16) is that of the target pixel.

$$\begin{matrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{matrix}$$

In all but the last line, and the last pixel of each line, of the starting image, all neighbors shown above exist. In those cases where some neighbors do not exist, the weighted average is taken over the target pixel and its existing neighbors.

3.3.2 Program EXPAND

Program EXPAND is the complement of Program COMPRESS. It solicits the image dimensions and then develops the output image by the algorithm shown below:

Input Pixel P in Row I, Column J

P	R
B	BR

Creates Output Pixels in Rows 2I and 2I+1, Columns 2J and 2J+1:

P	(P+R)/2
(P+B)/2	(P+R+B+BR)/4

where P is an input image pixel, R is P's right neighbor, B is P's neighbor below and BR is P's neighbor below and to the right. Pixels P comprise all pixels of the input image except those in the last row and column. This interpolation algorithm creates the expanded image except for its border pixels. Each border pixel is simply copied from the pixel immediately "inside" it, i.e., the pixel to the right of a left border pixel, below a top border pixel, etc.. The corner pixels are the same as their immediate diagonal neighbors, and, incidentally, horizontal and vertical neighbors.

3.3.3 Program SUB2IM

Program SUB2IM solicits the image dimensions and then subtracts one image from another, pixel by pixel. To each difference pixel SUB2IM adds a bias of 128 before storing the result in the output image to map the difference values into the range 0 to 255. If a biased result is outside this range (very unlikely if one input image is a reasonably good approximation of the other), it is clamped to the nearest end of the range.

3.3.4 Program ADD2IM

Program ADD2IM is the reverse of Program SUB2IM. It adds two images pixel by pixel and subtracts the bias of 128 before storing the result in the output image. Because of this assumed bias, one of the two images being added must be a "difference" image, or a coded (transmitted) approximation thereof, produced by SUB2IM.

4.0 Simulations

4.1 Test Image Selection Criteria

4.1.1 NCS Images

Several factors were involved in the selection of the test images employed in the simulations: image quality, availability and feature content. The NCS images are the four standard gray scale images developed for the NCS in a previous study cited in Section 1.2. The standard gray scale image selections were based on a set of characteristics designed to thoroughly test various gray scale transmission techniques.

Beyond the advantages these images provide in terms of image quality and availability, each image was selected because it contained several distinctive features that would aid in the subjective evaluation of the output images. The IEEE Face image is representative of an identification card. The House and Sky image contains large areas of gradually changing gray scale, several areas of varying texture, and various horizontal, vertical and diagonal lines. The House with Trees image is similar, but also contains high-detail regions, which make this image challenging with respect to coded image quality and compression. The Aerial Photograph is a low contrast image of high detail and relatively low resolution.

4.1.2 JPEG Images

Four JPEG images, "Barb2," "Boats," "Zelda" and "Balloons," were selected because, in JPEG simulations, "Barb2" gave the worst compression, "Balloons" gave the best, and the other two gave intermediate values.

4.2 Simulation Selection Criteria

Three transmission modes were simulated: sequential, hierarchical progression and "bit slicing" in the image domain.

Sequential transmission, hierarchical progression and (transform-domain) bit slicing are defined in Sections 2.6 and 2.8. Image-domain "bit slicing" consists of sequentially transmitting an image with a quantization scale factor of 200, and then sequentially transmitting difference "images" with scale factors of 100, 50 and 25. This is similar to transform-domain bit slicing in that the quantum numbers for the difference images are always 0, plus 1 or minus 1, corresponding to the 1-bit corrections in the transform-domain bit slicing progression. This fact is proven in Appendix A. Image-domain bit slicing should therefore give the same coded images as JPEG bit slicing, but worse compression because it does not retain quantum number "history." Appendix A shows that this is the case except that the coded images are not necessarily identical because of the rounding rules for negative numbers. Image-domain "bit slicing" was simulated to determine whether it could serve as a "poor man's" bit slicing scheme, because it is simpler and requires less memory than transform-domain bit slicing. Unfortunately, the compression degradation was too severe to make this approach a practical option.

Sequential transmission, the core of the proposed baseline system standard, was simulated for all four NCS images with quantization scale factors of 200, 100, 50 and 25. These scale factors were chosen for two reasons: (1) to cover the range from poor image quality and high compression to good image quality and low compression; and (2) to match the scale factors explicit to image-domain "bit slicing" and implicit to transform-domain bit slicing wherein the quantum numbers obtained with a scale factor of 25 are divided by 8, 4, 2 and 1 in the four transmission passes. Sequential transmissions of the four JPEG images (luminance only) were also simulated with a quantization scale factor of 25, and the results were compared with JPEG results.

Hierarchical progression and "bit slicing" in the image domain were simulated for the House with Trees image only, so that the results of different transmission modes could be compared for the same image. The House with Trees image was chosen because it is the most difficult of the four NCS images to compress.

4.3 Results

4.3.1 Sequential Transmission

4.3.1.1 NCS Standard Gray Scale Images

Table 4.1 shows the results of simulating sequential transmission of the four NCS images. The Subjective Image Quality Rating is defined as follows:

Constant Parameters

Image Resolution:	200 pixels per inch
Image Size:	512 X 512 pixels
Block Size:	8 X 8 pixels
Transmission Mode:	Sequential

Image	Quantization Scale Factor	Compressed Bits per Pixel	RMS Error	Subjective Image Quality Rating
IEEE Face (Figs. 5.2 - 5.5)	25	0.72	1.54	10
	50	0.47	2.16	9
	100	0.31	3.14	8
	200	0.20	4.77	6
House and Sky (Figs. 5.7 - 5.10)	25	0.68	1.85	10
	50	0.44	2.50	9
	100	0.27	3.45	8
	200	0.17	4.96	6
House with Trees (Figs. 5.12 - 5.15)	25	1.48	3.92	10
	50	1.02	5.41	9
	100	0.68	7.33	8
	200	0.41	10.08	6
Aerial Photograph (Figs. 5.17 - 5.20)	25	1.12	2.07	10
	50	0.79	3.02	10
	100	0.53	4.50	9
	200	0.34	6.80	7

Table 4.1 Performance and Compression of NCS Standard Gray Scale Images

Rating	Definition
0	Image is not recognizable.
1	Almost no detail is evident; only general outlines of objects remain.
2	Loss of edge detail almost total; objects in image unrecognizable.
3	Image is slightly recognizable; edge boundaries severely distorted.
4	Image is partially recognizable; complete loss of detail in several image regions is evident.
5	Image is recognizable, but shows severe blocking artifact throughout and poor detail rendition.
6	Blocking is severe in regions of high detail, and detail rendition is marginal.
7	Blocking is moderate in high-detail areas, and detail rendition is fair.
8	Blocking is slightly evident, and detail rendition is good.
9	No blocking is evident, and detail rendition is very good.
10	Encoded image is indistinguishable from original image.

Figure 4.1 shows log-log plots of the bit rates and RMS errors for the same four images. The points lie almost on straight lines. Thus, to very good approximation, the bit rate and the RMS errors can be expressed by $\log y = a + b \log S$, where y is bit rate or RMS error, S is the quantization scale factor, and a and b are constant for a given image.

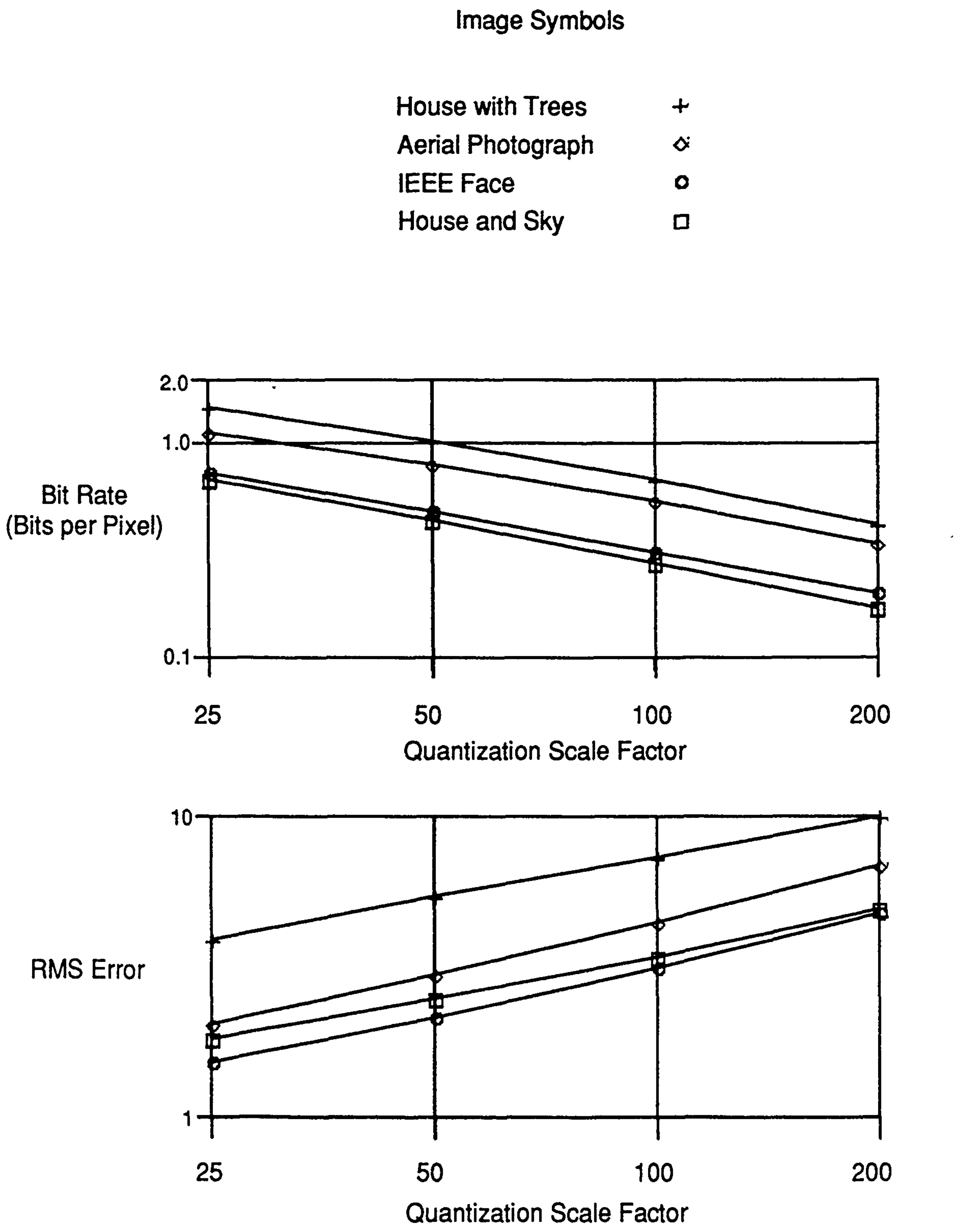


Figure 4.1 Log-log Plots of Bit Rate and RMS Error vs. Quantization Scale Factor for Sequential Transmission

Doc. N800 shows this approximation for the bit rate, and suggests using it to control data compression. Unfortunately, at least two iterations consisting of measuring compression for a given S value are required to obtain the values of a and b, and thence the value of S required for the desired compression. Because such an iterative procedure requires large amounts of computation, JPEG has since abandoned this suggestion in favor of selecting a value of S to give approximately the desired tradeoff between data compression and image quality.

4.3.1.2 JPEG Images

Sequential transmissions of the luminance parts of four JPEG images were simulated with a quantization scale factor of 25 (high quality, relatively low compression). Figure 4.2 shows a comparison of the study and published JPEG results.

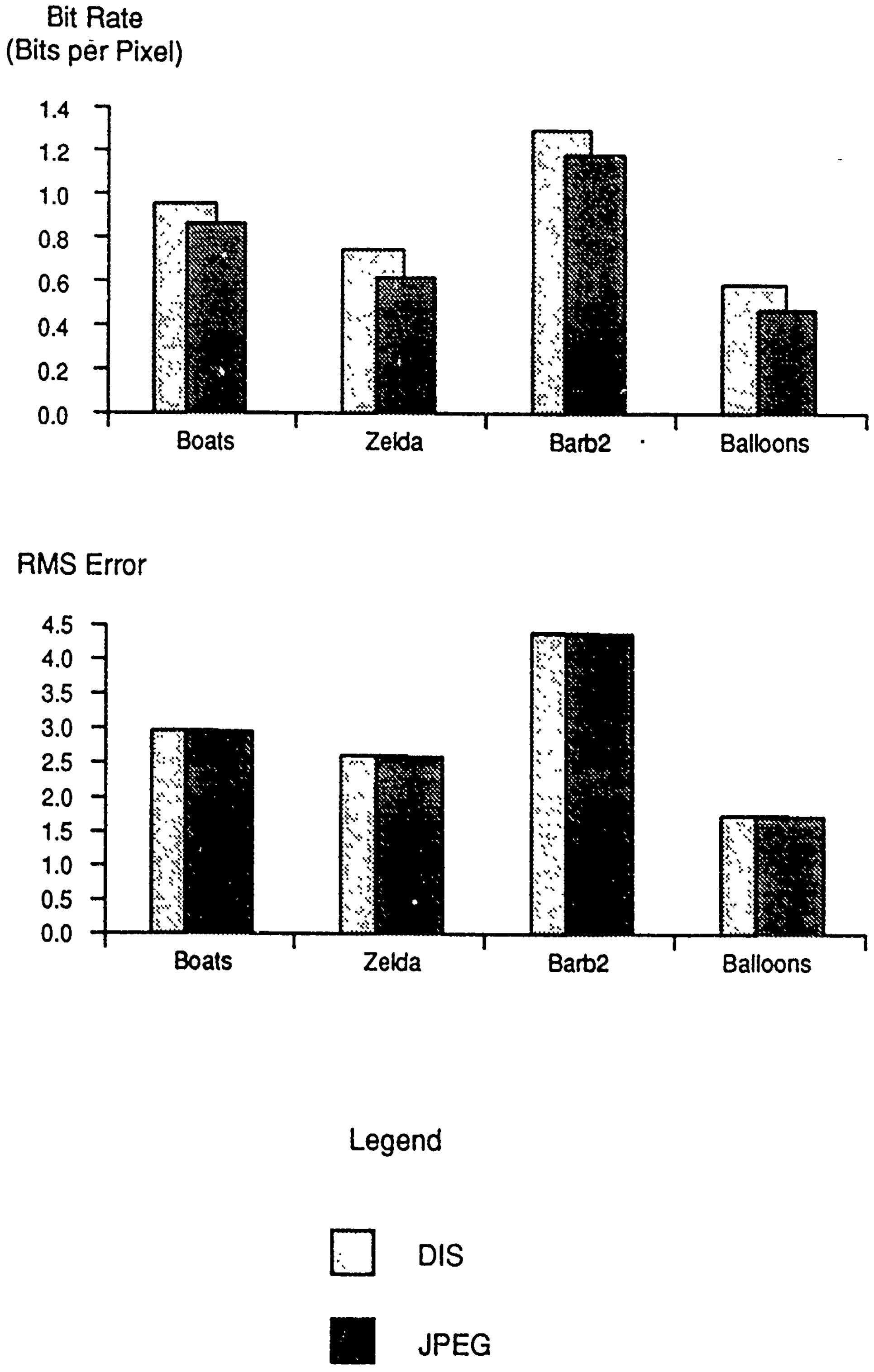
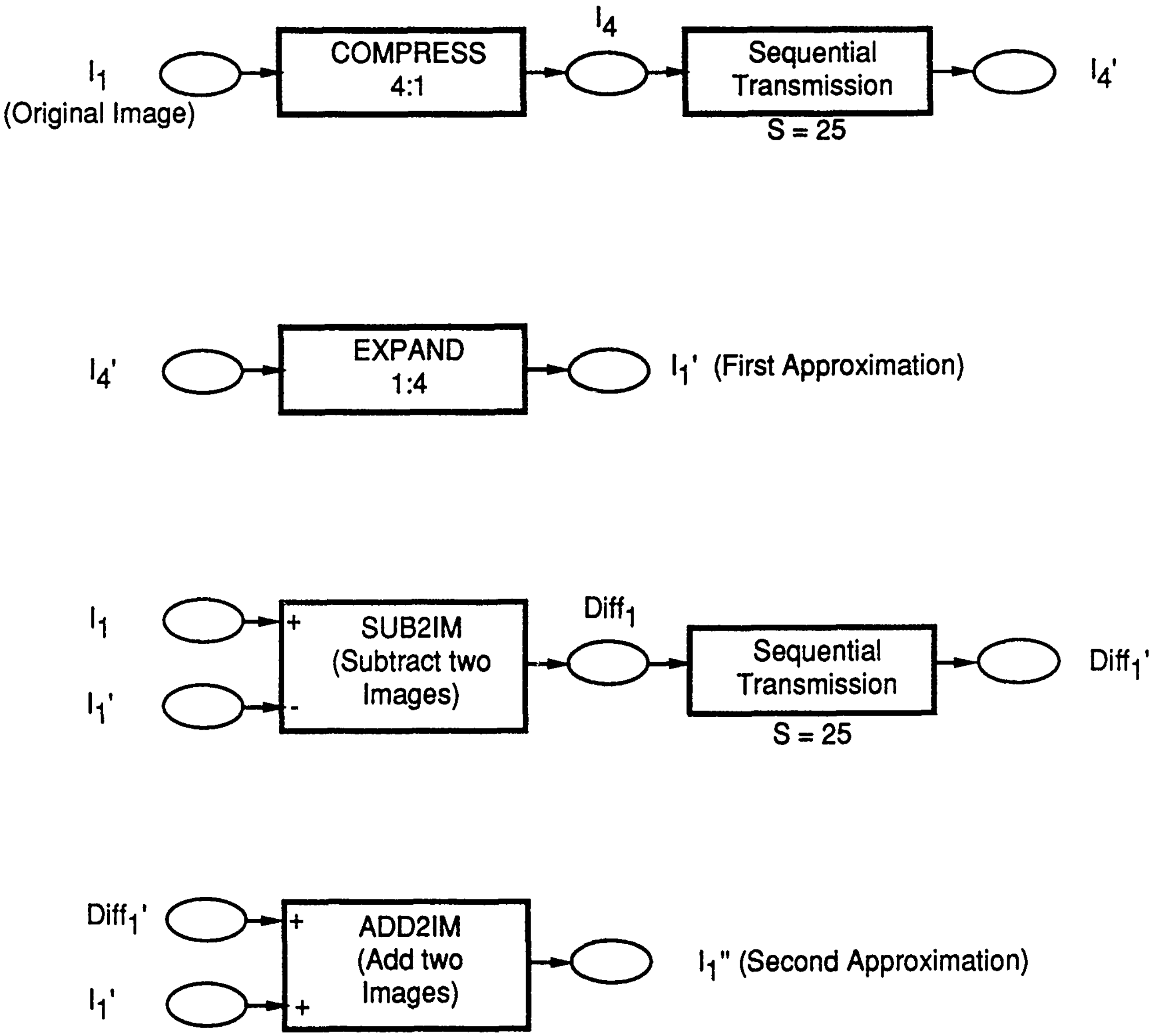


Figure 4.2 Comparison of DIS and JPEG Sequential Transmissions with Quantization Scale Factor of 25

The RMS errors were identical to within tenths of a percent. Thus the Simulation and JPEG systems produce nearly identical results except for data compression. Because the Simulation System employs a simpler model for the Q coder, the study bit rates were consistently greater than the JPEG rates. The differences ranged from 0.086 to 0.123 bits per pixel for the four images. The largest difference in the bit rate was only 1.4 times the smallest, even though the JPEG bit rates ranged from 0.476 to 1.193, the worst 2.5 times the best. Thus, the absolute difference was roughly constant (0.1 bit per pixel), and the percentage difference was greatest for the smallest bit rate.

4.3.2 Hierarchical Progression

Figure 4.3 shows a two-pass hierarchical progression in which the quantization scale factor was 25 in both passes. In the first pass a sub-sampled version of the original image (one-fourth the number of pixels) was sequentially transmitted. The received image was interpolated to full size, and the difference between the original and interpolated images was transmitted in the second pass. The results are shown in Figure 4.4.



S = Quantization Scale Factor

Figure 4.3 Two-Pass Hierarchical Progression

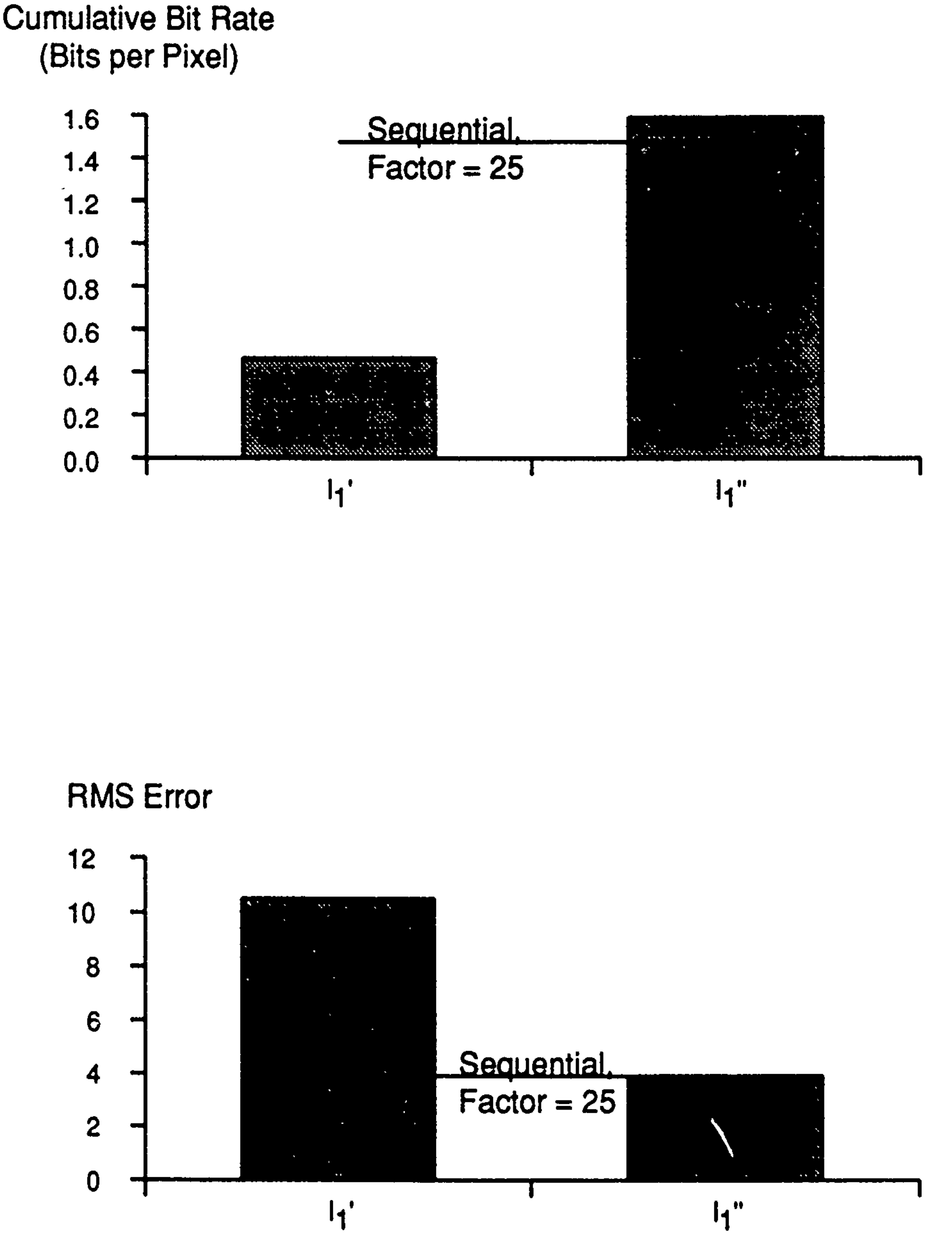
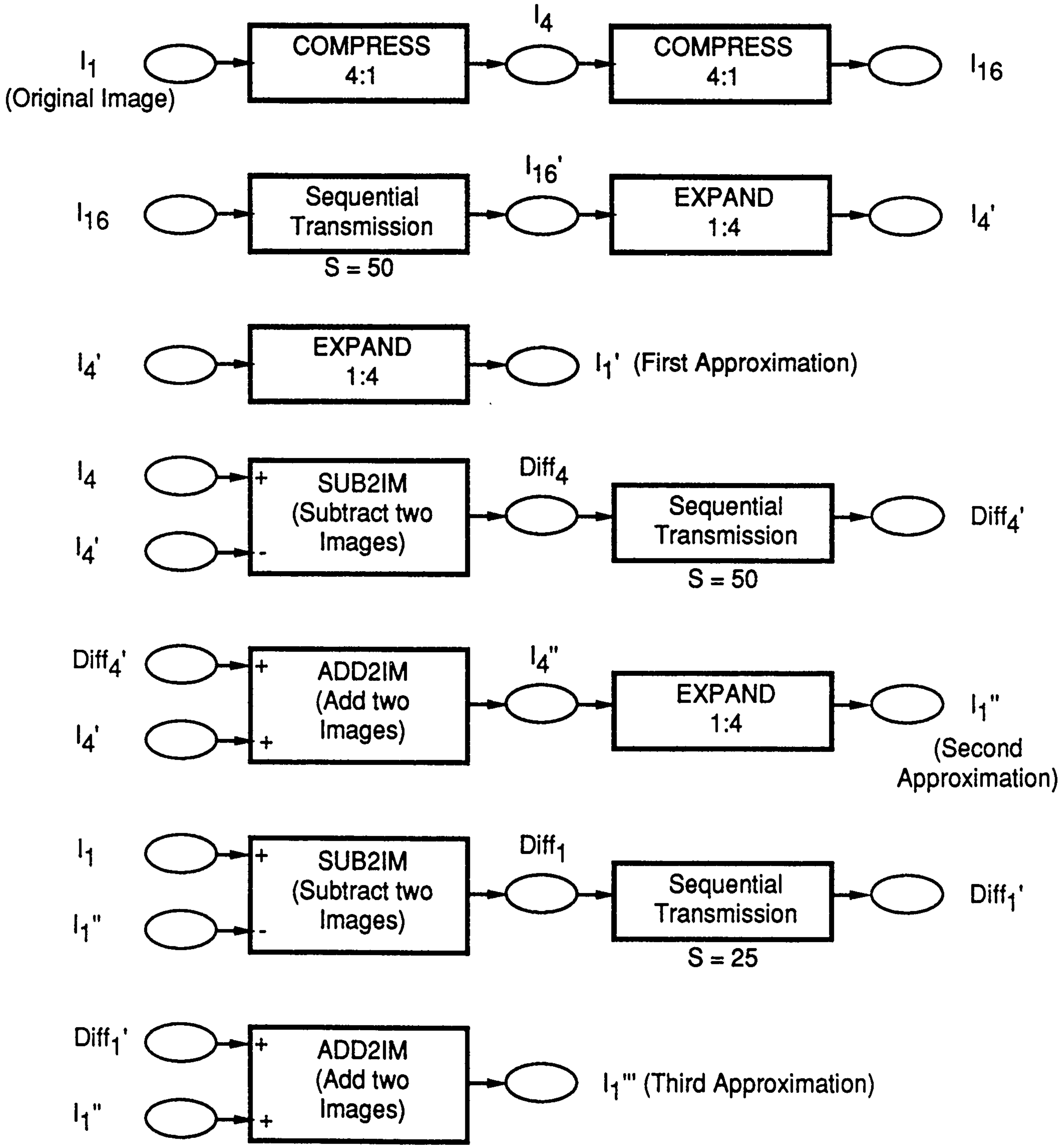


Image: House with Trees

Figure 4.4 Results of Two-Pass Hierarchical Progression Shown in Figure 4.3

The RMS error after the second pass was 2.6 percent greater than that of the sequential transmission, and the accumulated bit rate was 7.7 percent greater.

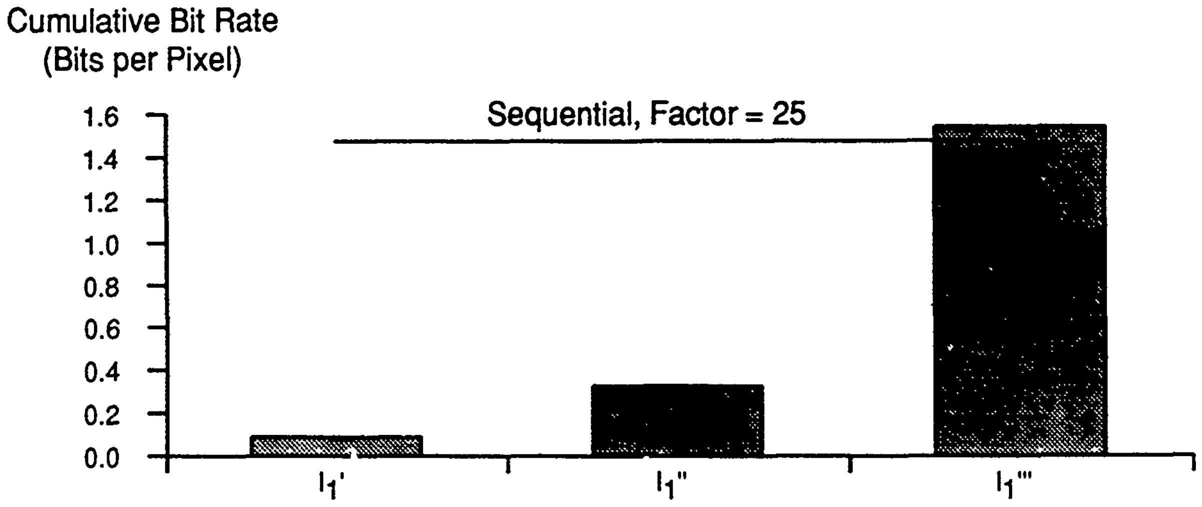
Figure 4.5 shows a three-pass hierarchical progression similar to the first three passes shown in Doc. N800. The quantization factor was chosen to be 50 for the first two passes and 25 for the final refinement. The choice of 50 was made for the first two passes to give effective bit rates (bits per pixel of the full-sized image) roughly similar to those specified in Doc. N800. The choice of 25 for the last transmission was made for compatibility with the sequential transmission with which the results were compared.



S = Quantization Scale Factor

Figure 4.5 Three-Pass Hierarchical Progression

Figure 4.6 shows the results. The final RMS error was 3.0 percent greater than that of the sequential transmission, and the accumulated bit rate was 4.1 percent greater. Thus, the three-pass hierarchical progression gave approximately 3 percent better compression and 0.4 percent worse RMS error than the two-pass hierarchical progression.



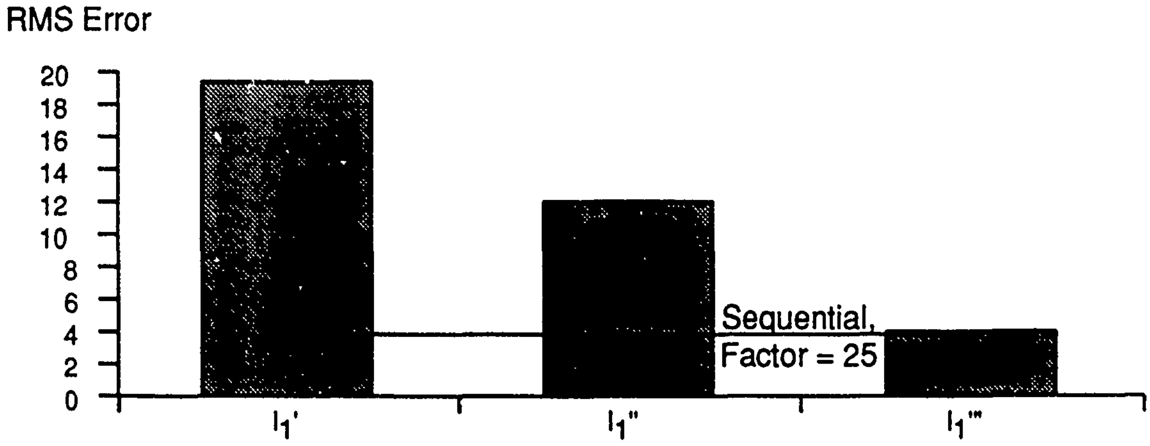


Image: House with Trees

Figure 4.6 Results of Three-Pass Hierarchical Progression Shown in Figure 4.5

Figure 4.7 shows the results of a four-pass hierarchical progression. This was the same as the three-pass progression, except the third pass was transmitted with a quantization scale factor of 50 instead of 25, and a final refinement was transmitted with a scale factor of 25.

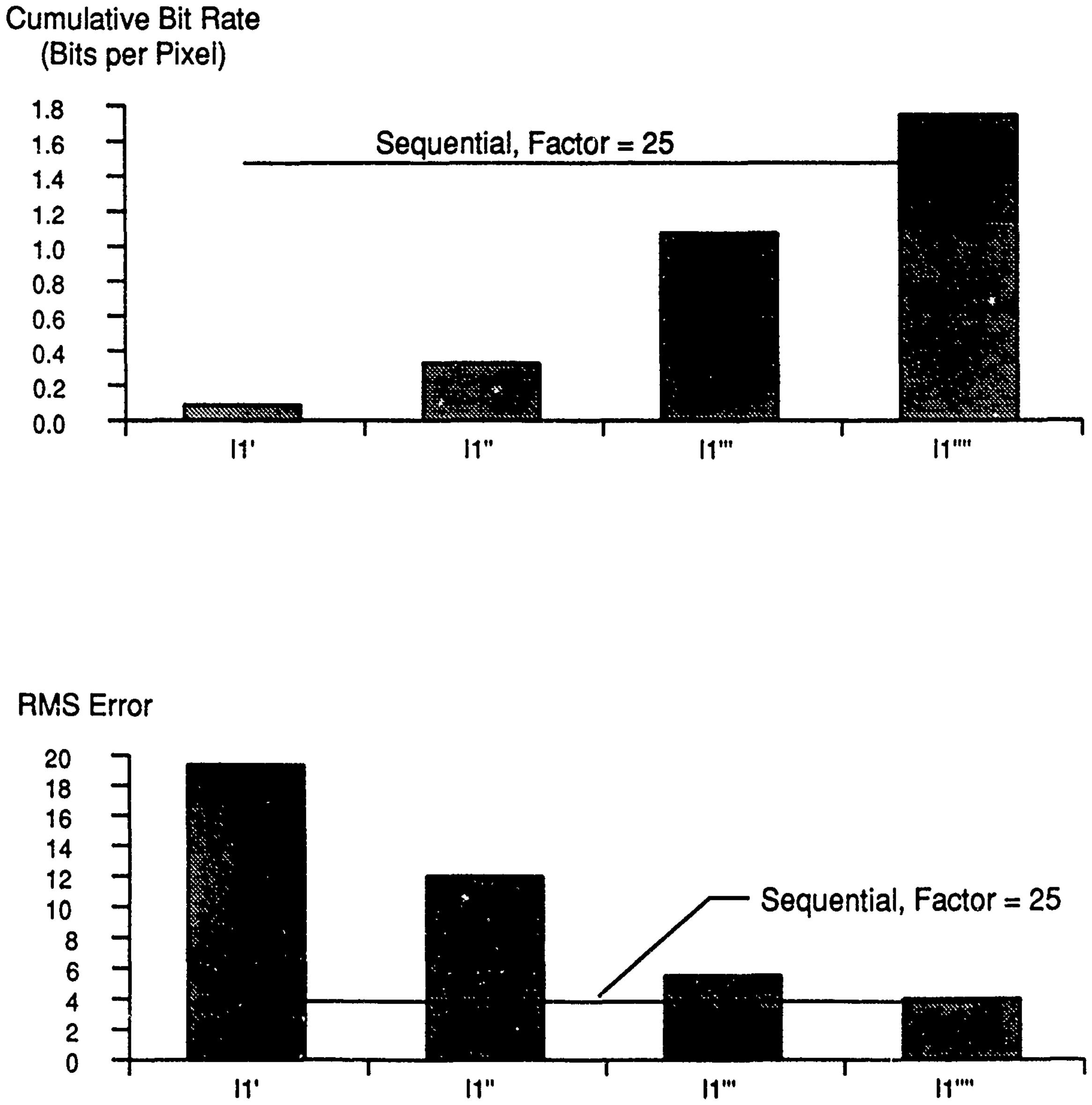


Image: House with Trees

Figure 4.7 Results of Four-Pass Hierarchical Progression

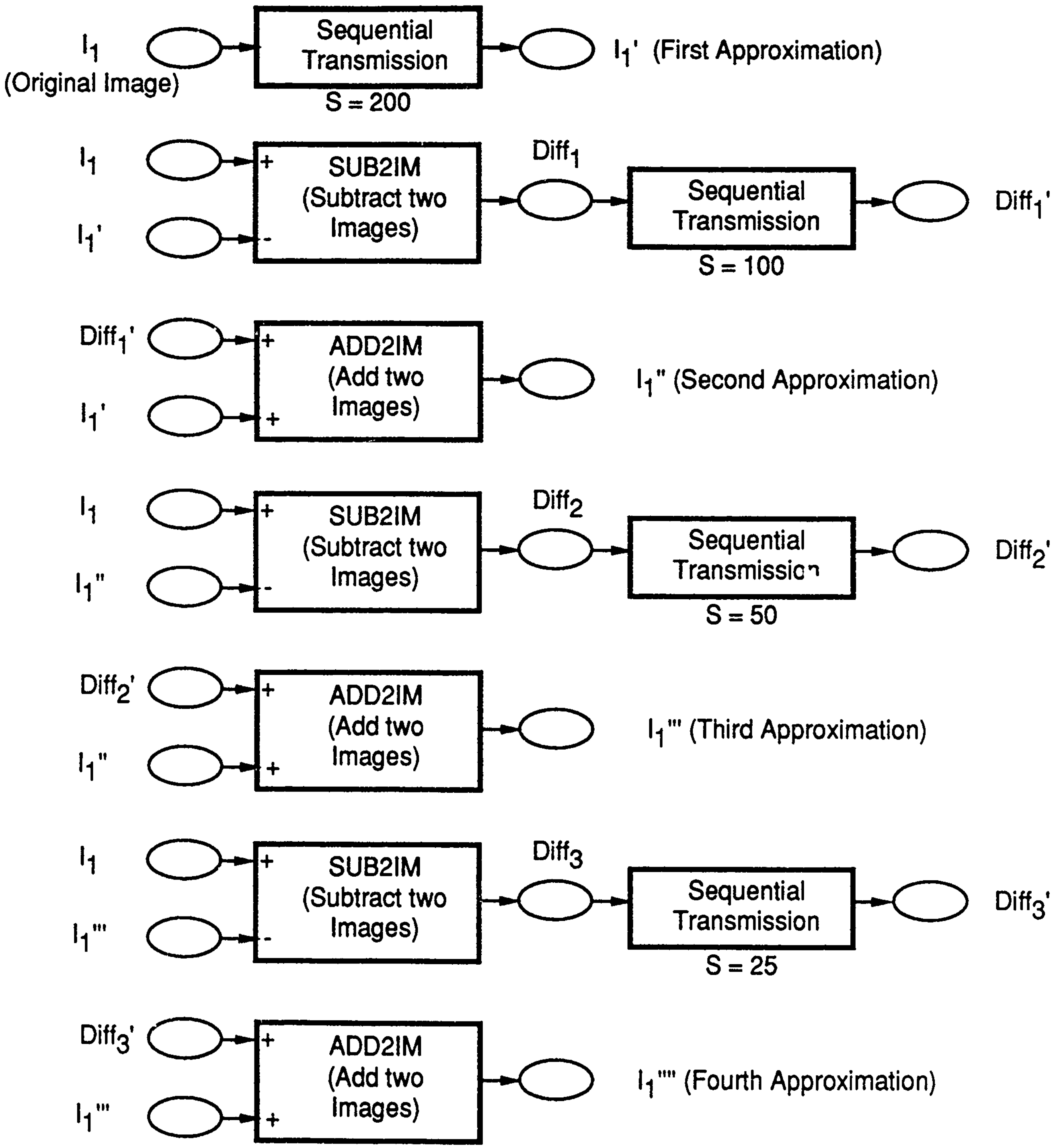
This progression is identical to that specified at the beginning of Doc. N800 except for somewhat different bit rates, owing to different choices of quantization scale factors. The final RMS error was 3.1 percent greater than that of the sequential transmission, and the accumulated bit rate was 18 percent greater. Moreover, the bit rate was 13 percent greater than for the three-pass case.

Four passes giving significantly worse compression than three is attributed to the fact that the fourth pass refined a more coarsely quantized image instead of an interpolated image, i.e., the fourth pass acted like the final pass of an image-domain "bit slicing" progression. Appendix A offers some insight as to why, as results to be presented next show, image-domain "bit slicing" gives worse compression than hierarchical compression. It is concluded that in any hierarchical progression, whether sub-sampling occurs once or more than once, the progression should start by transmitting the most deeply sub-sampled image. The coded image should then be interpolated by both transmitter and receiver to the next higher sub-sampling level. The difference between the actual and interpolated image at this new level should then be transmitted. This process is continued until a refinement is transmitted at full size. At <u>no</u> time should more than one refinement be transmitted for the same sub-sampling level. In other words, a fairly high bit rate per <u>transmitted</u> pixel should always be used to avoid having to refine an image to correct for coarse quantization.

The RMS errors of the final images in all three hierarchical progressions were slightly greater than that for the sequential transmission, and the greater the number of passes, the greater the difference. However, the worst (four-pass) RMS error was only 3.1 percent greater than for the sequential case; hence, it is concluded that if the final refinement is made with the same quantization scale factor as in the sequential transmission, the difference between the two final images will be indistinguishable. The effect of hierarchical transmission on RMS error is discussed further in Appendix A.

4.3.3 "Bit Slicing" in the Image Domain

Figure 4.8 shows the image-domain "bit slicing" transmission simulation, and Figure 4.9 shows the results compared with those of sequential transmissions with the same quantization scale factors.



S = Quantization Scale Factor

Figure 4.8 "Bit Slicing" in the Image Domain

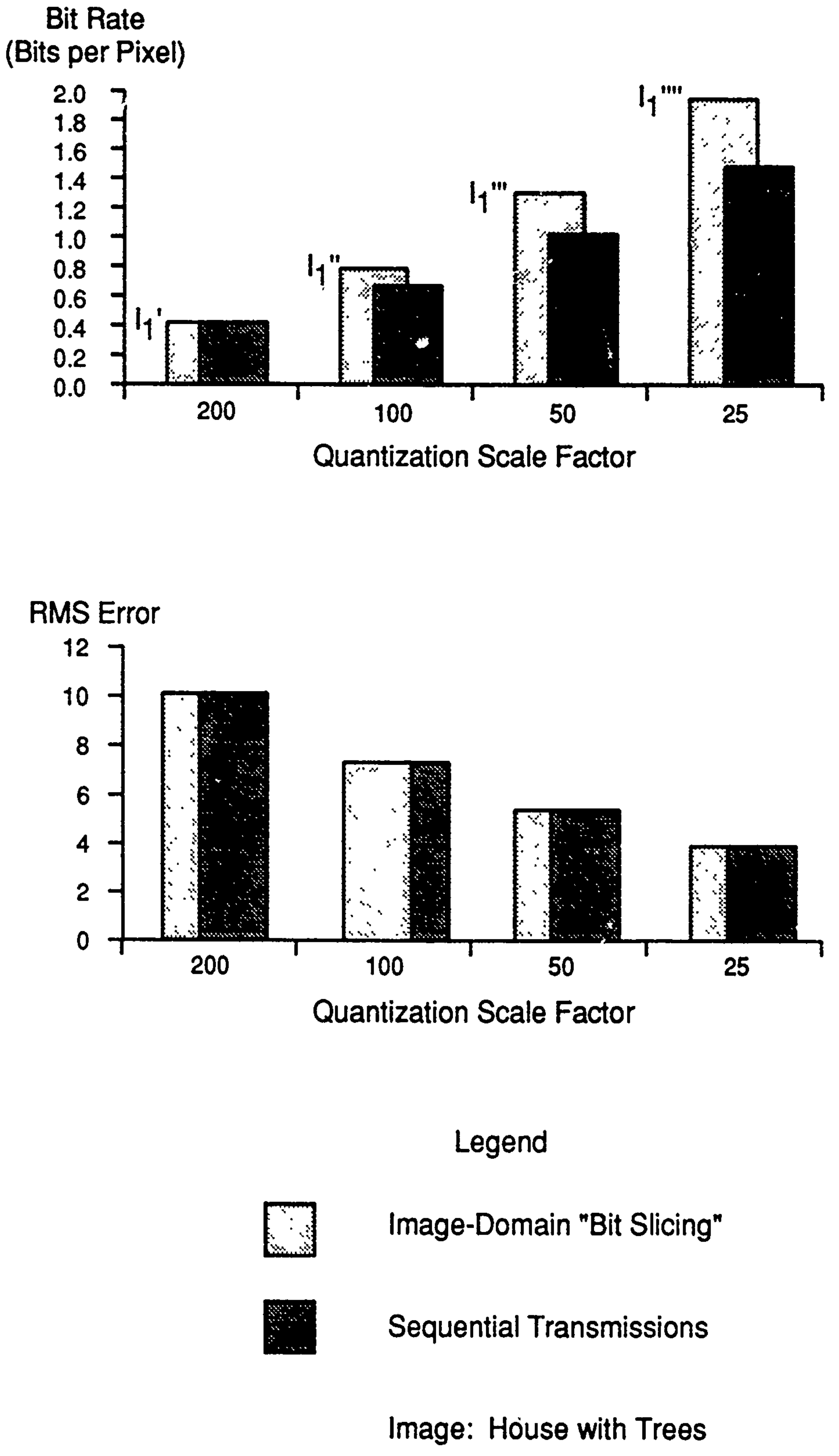


Figure 4.9 Comparison of Image-Domain "Bit Slicing" (Figure 4.8) with Sequential Transmissions

The first pass of the progression was the sequential transmission with a scale factor of 200; hence the compression and RMS error are obviously the same as in the sequential case. In succeeding refinements, however, the compression deteriorated badly with respect to the corresponding sequential transmissions. This contrasts with transform-domain bit slicing, in which the compression was slightly better than with sequential transmissions, as discussed in Section 2.8.2.

This deterioration of compression with number of refinements led to the conclusion stated above, namely, that, in a hierarchical progression, no more than one refinement should be transmitted at a given sub-sampling level.

The RMS errors were very slightly (tenths of a percent) worse than those of the sequential transmissions, the differences increasing with number of passes. As is shown in appendix A, these tiny differences in RMS error are attributed to the rounding rules for negative numbers. In transform-domain bit slicing, the final image is guaranteed to be identical to that produced by a sequential transmission with a scale factor of 25, because, after the final refinement, the coefficient quantum numbers are the same.

4.3.4 Subjective Evaluation

Table 4.2 is an index to photographs showing the results of simulating sequential transmissions of all four NCS images. Subjective image quality ratings for these images are shown in

Table 4.1 earlier in this section. Table 4.3 is a similar index for hierarchical progression simulations for the House with Trees image and includes subjective image quality ratings. Photographs of image-domain "bit slicing" images are not shown, because these images are nearly identical to those produced by sequential transmissions.

Image	Quantization Scale Factor	Image Figure Number
IEEE Face	(Original)	4.10
	25	4.11
	50	4.12
	100	4.13
	200	4.14
House and Sky	(Original)	4.15
	25	4.16
	50	4.17
	100	4.18
	200	4.19
House with Trees	(Original)	4.20
	25	4.21
	50	4.22
	100	4.23
	200	4.24
Aerial Photograph	(Original)	4.25
	25	4.26
	50	4.27
	100	4.28
	200	4.29

Table 4.2 Image Index for Sequential Transmissions

Image: House withTrees

Number of Passes	Pass	Image Transmitted	Quantization Scale Factor	Image Displayed	Subjective Image Quality Rating	Image Figure Number
2	1	I_4	25	I_1'	6	4.30
	2	$I_1 - I_1'$	25	I_1''	9	4.31
3	1	I_{16}	50	I_1'	4	4.32
	2	$I_4 - I_4'$	50	I_1''	6	4.33
	3	$I_1 - I_1''$	25	I_1'''	9	4.34
4	1	I_{16}	50	I_1'	4	4.35
	2	$I_4 - I_4'$	50	I_1''	6	4.36
	3	$I_1 - I_1''$	50	I_1'''	9	4.37
	4	$I_1 - I_1'''$	25	I_1''''	10	4.38

Table 4.3 Image Index for Hierarchical Progressions

Figures 4.10 through 4.29 illustrate the effects of the quantization scale factor (QSF) on the sequential transmission of gray scale images. With QSF=25, virtually no image degradation is evident in any of the four NCS test images processed (see Figures 4.11, 4.16, 4.21, and 4.26). Only slight blocking artifacts are evident in low detail image regions in the simulations in which QSF=50 was employed (see Figures 4.12, 4.17, 4.22, and 4.27). With QSF=100, (Figures 4.13, 4.18, 4.23, and 4.28) blocking is perceptible in the low to medium detail image regions, along with a noticeable loss of high detail. With QSF=200, blocking is prevalent in all image regions, and significant medium to high detail loss is evident.

Figures 4.30 through 4.38 illustrate the effects of the subsampling and interpolation processes associated with the transmission of gray scale images using hierarchical progression. Figures 4.30 and 4.31 illustrate a two-pass progression in which 4-to-1 subsampling was employed. Figure 4.30 illustrates the reconstructed image after the 1-to-4 interpolation was applied; the image appears blurred, and a loss of high detail (e.g. roof tiles, leaf/branch detail) is evident. In Figure 4.31, a difference image between the original and interpolated images was employed to enhance the interpolated image; the blurring is virtually eliminated, and the high detail regions are much sharper.

Figures 4.32, 4.33, and 4.34 illustrate a three-pass progression in which 16-to-4-to-1 subsampling was employed.

Figure 4.32 illustrates the reconstructed image after the 1-to-4-to-16 interpolation was applied; the image appears extremely blurred, and a loss of almost all medium to high detail is evident. In Figure 4.33, a difference image between the original 4-to-1 subsampled image and the 1-to-4 interpolated image was employed prior to the 4-to-16 interpolation step to enhance the interpolated image; the blurring is significantly reduced, and much more detail is evident. In Figure 4.34, a difference image between the original image and the enhanced interpolated image from pass 2 was employed to enhance the interpolated image further; the blurring is virtually eliminated, and the high detail regions are much sharper.

Figures 4.35 through 4.38 illustrate a four-pass progression in which 16-to-4-to-1 subsampling was employed. This progression is basically the same as the three-pass progression except that an additional difference enhancement step was employed to further improve the interpolated image quality. The first two passes of the four-pass progression are identical to those of the three-pass progression; in the third pass, a coarser quantization scale factor is employed in the transmission of the difference image (between the original image and the enhanced interpolated image from pass 2). The resulting twice enhanced interpolated image, displayed in Figure 4.37, contains slightly more blurring and exhibits a marginally larger loss of detail than the image produced in pass 3 of the three-pass progression. In Figure 4.38, a difference image between the original image and the twice enhanced interpolated image from pass 3 was employed to enhance the interpolated image further; the resulting image is virtually identical to the original image.

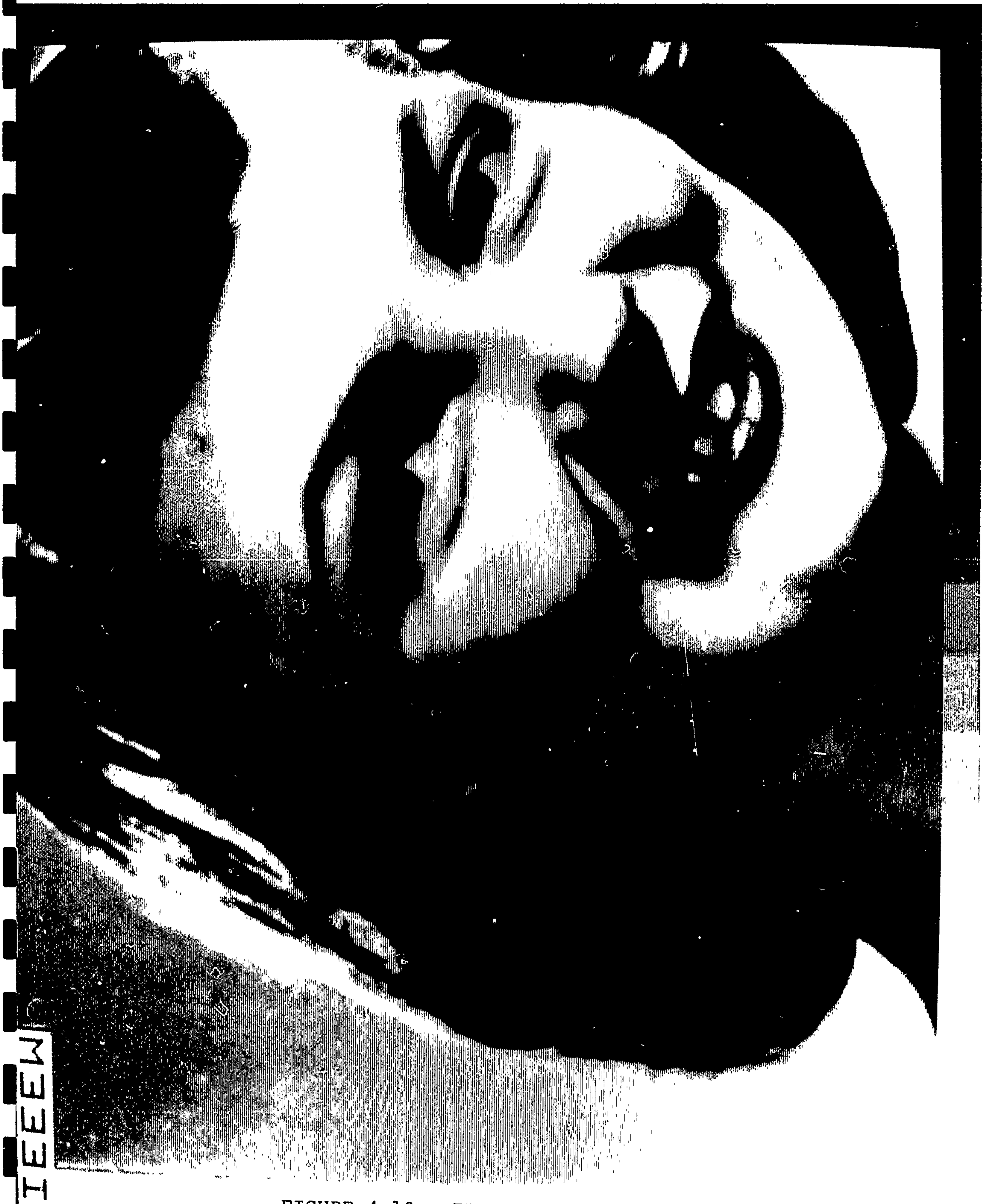


FIGURE 4.10 - IEEE FACE, ORIGINAL IMAGE

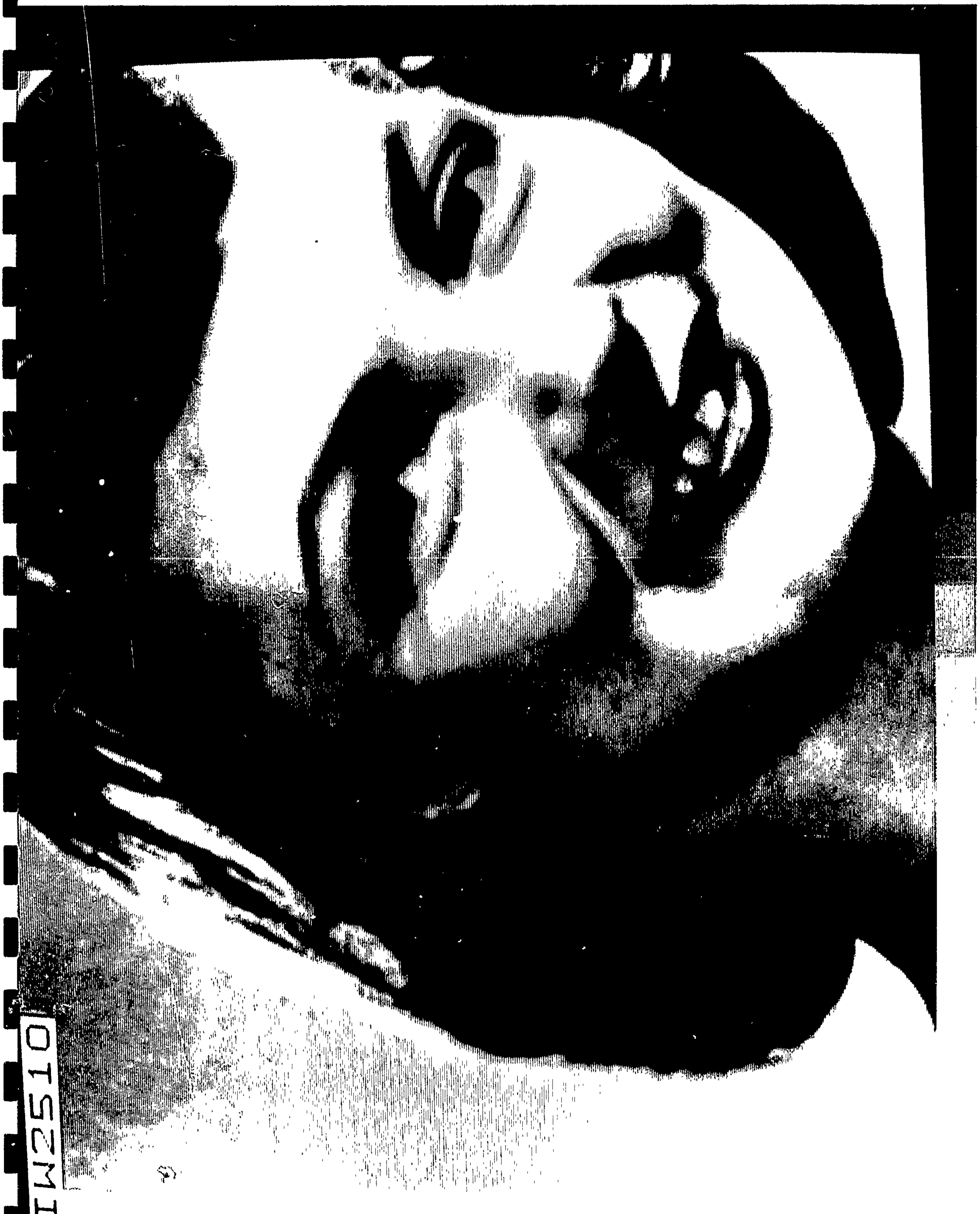


FIGURE 4.11 - IEEE FACE, QSF = 25

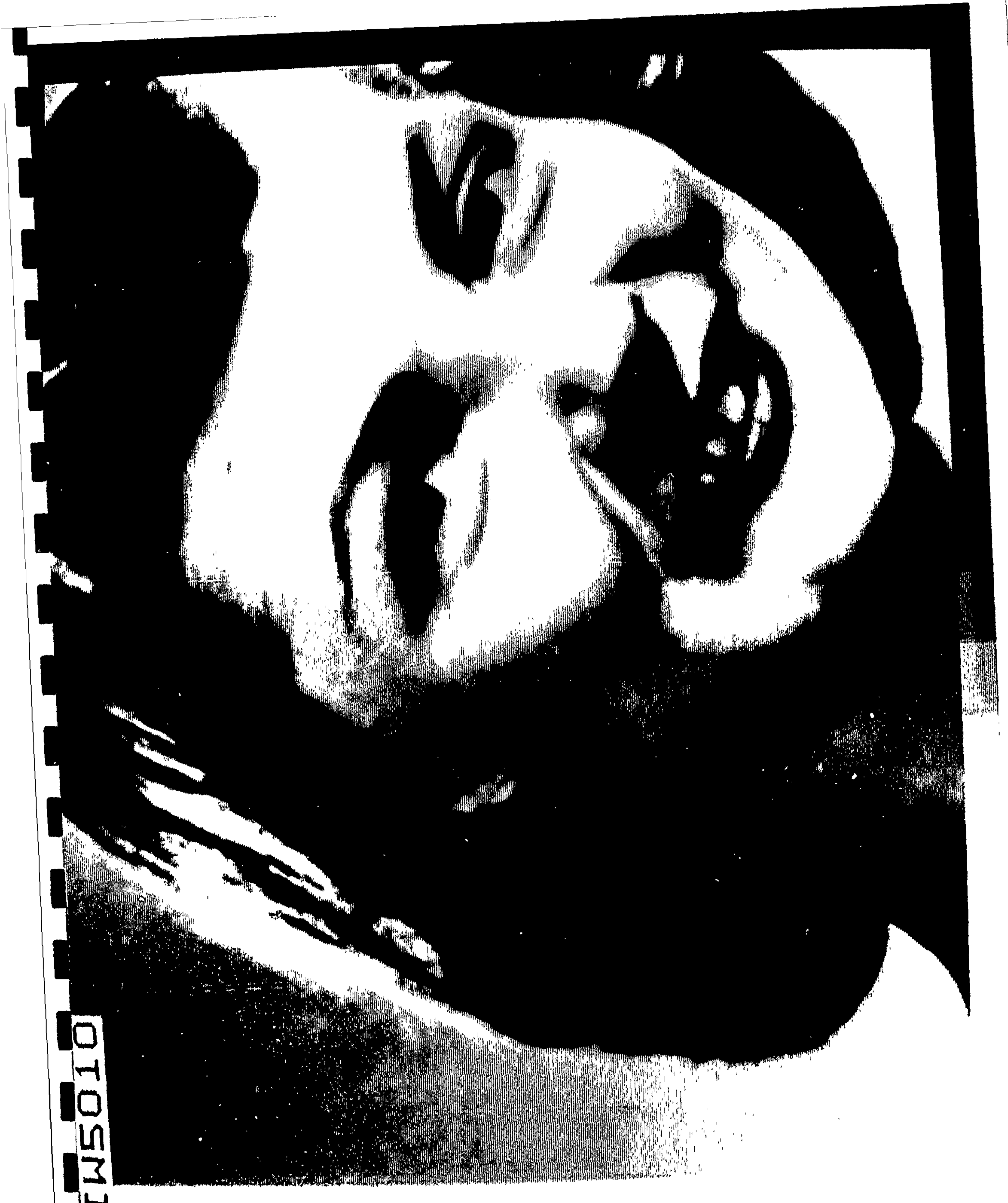


FIGURE 4.12 - IEEE FACE, QSF = 50

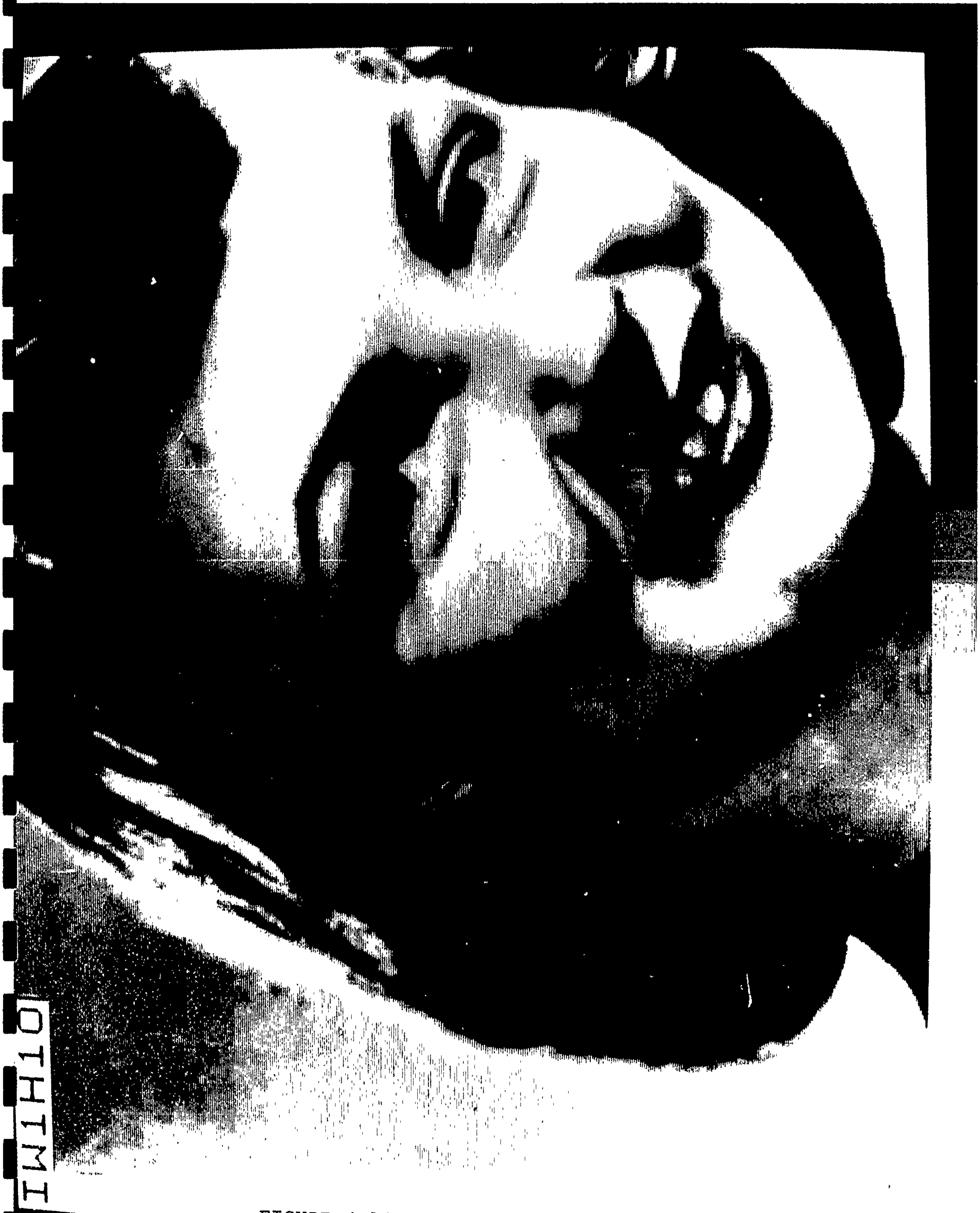
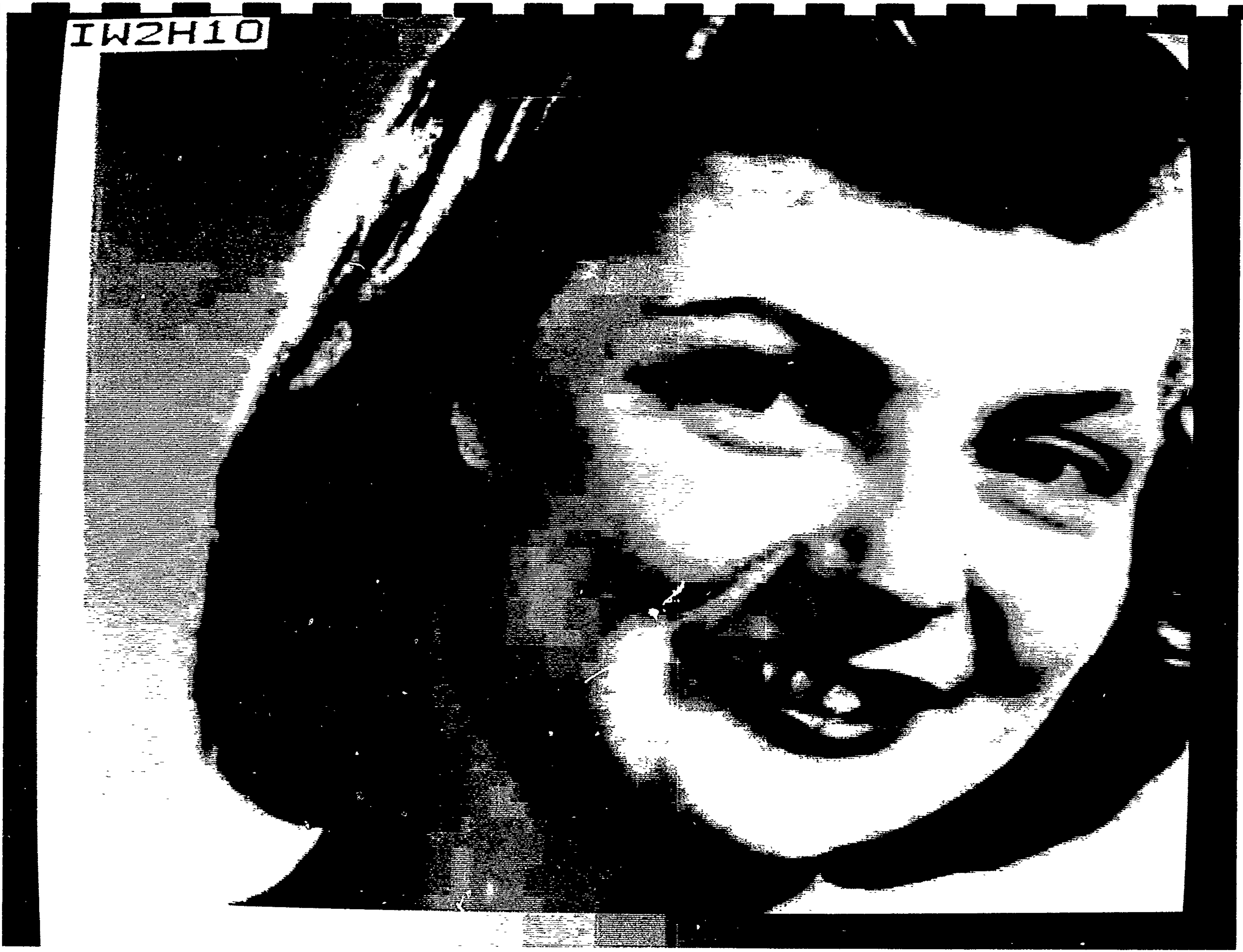


FIGURE 4.13 - IEEE FACE, QSF = 100


IW2H10

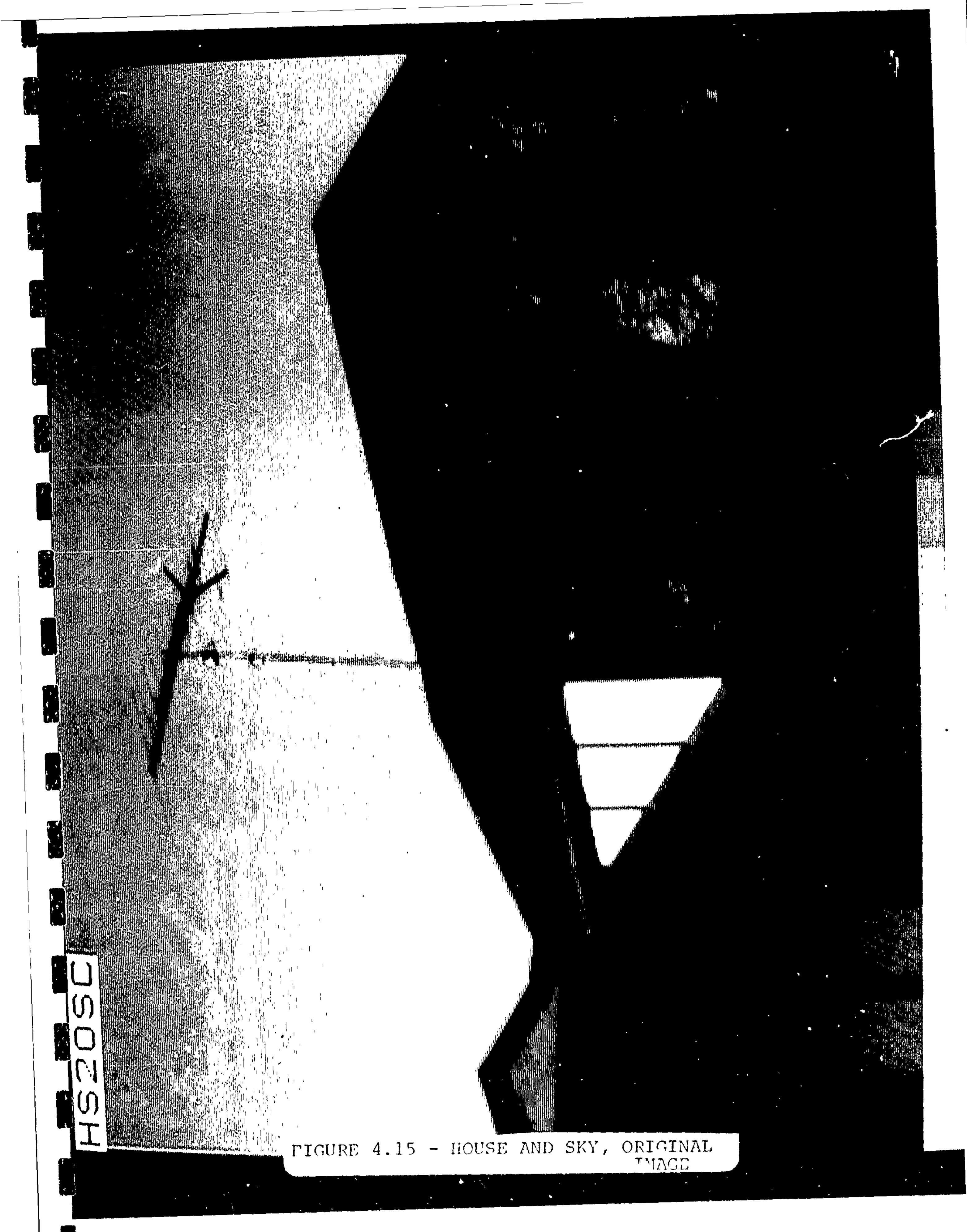


FIGURE 4.15 - HOUSE AND SKY, ORIGINAL IMAGE

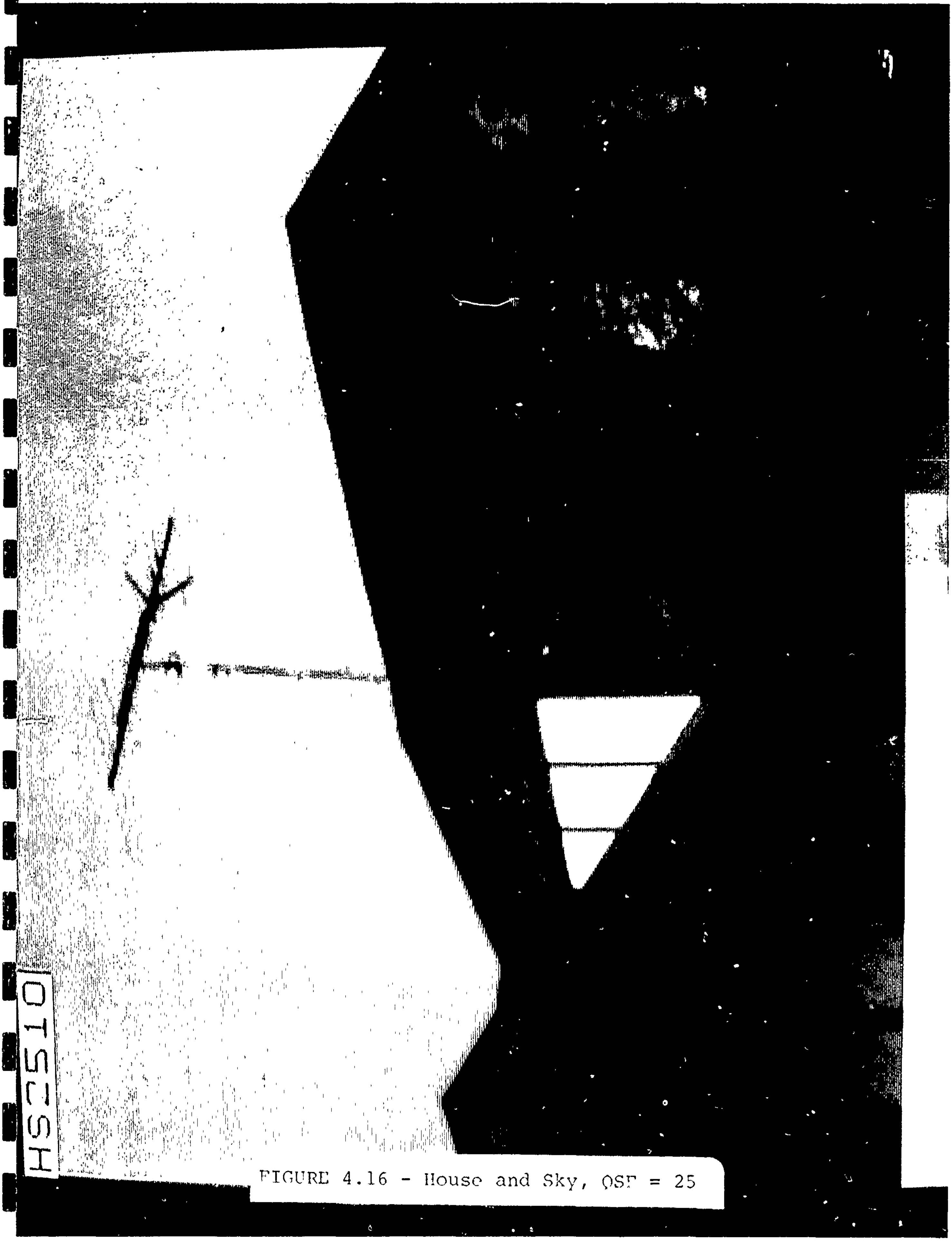


FIGURE 4.16 - House and Sky, QSF = 25

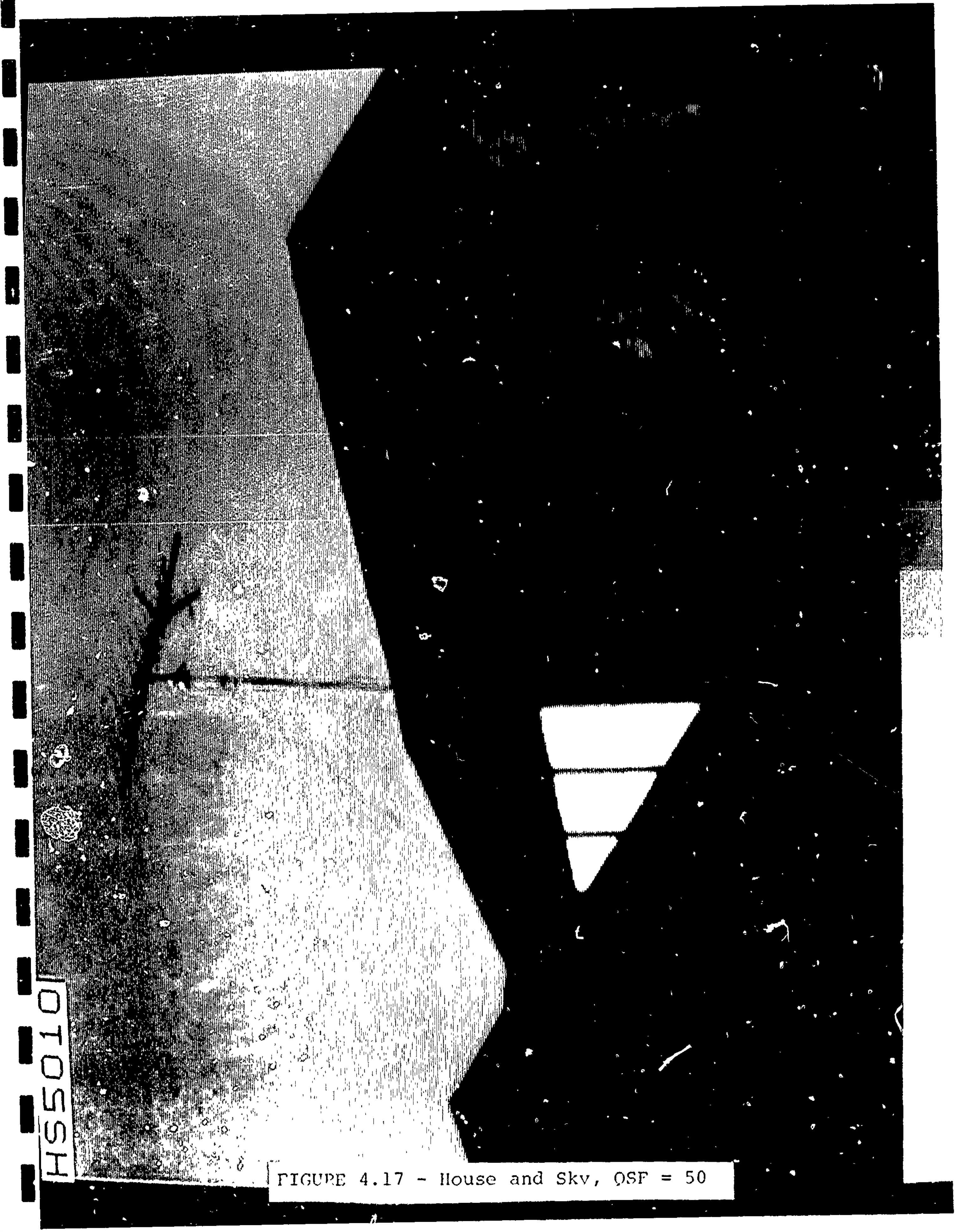


FIGURE 4.17 - House and Sky, OSF = 50

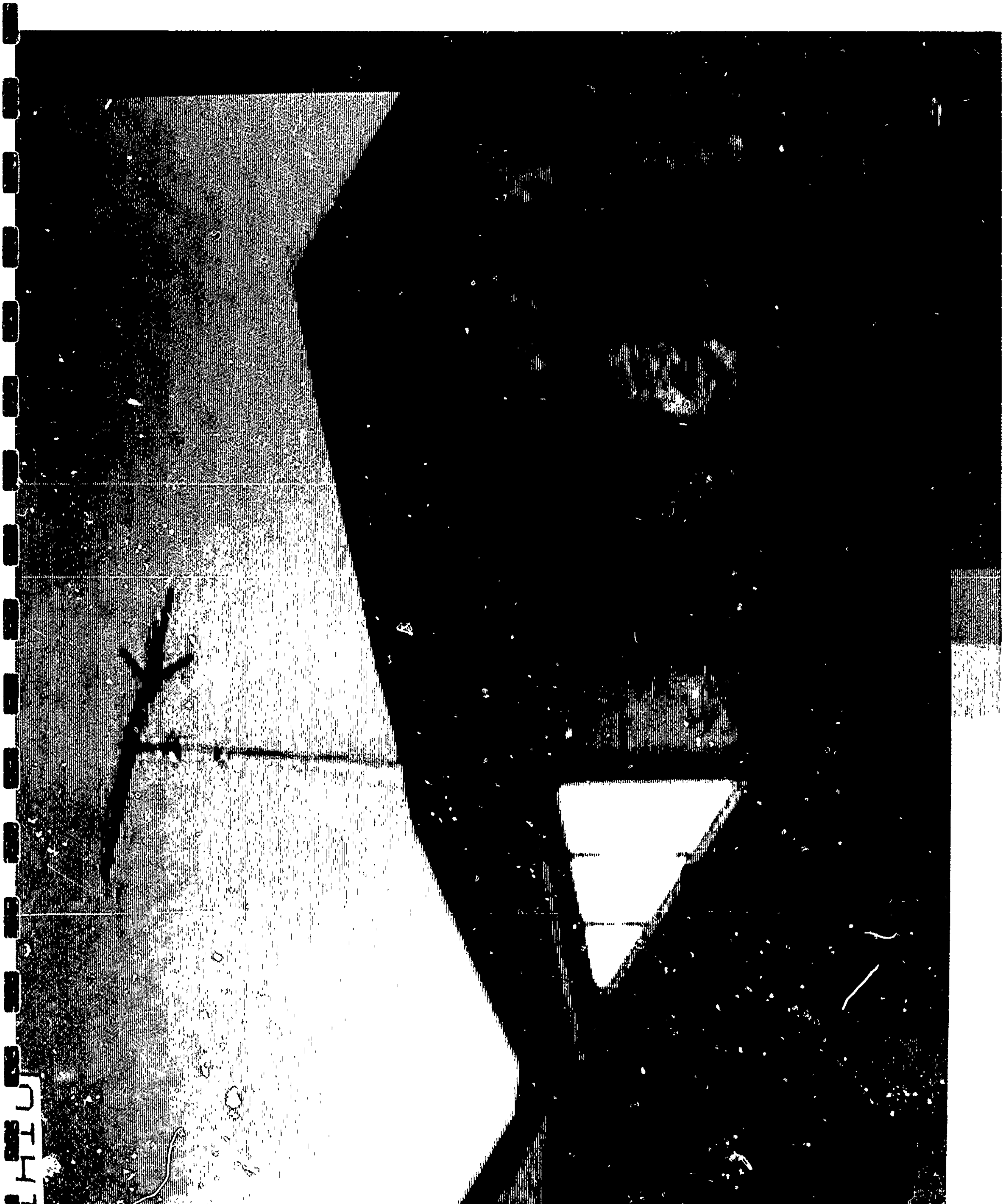


FIGURE 4.18 - House and Sky, QSF = 100

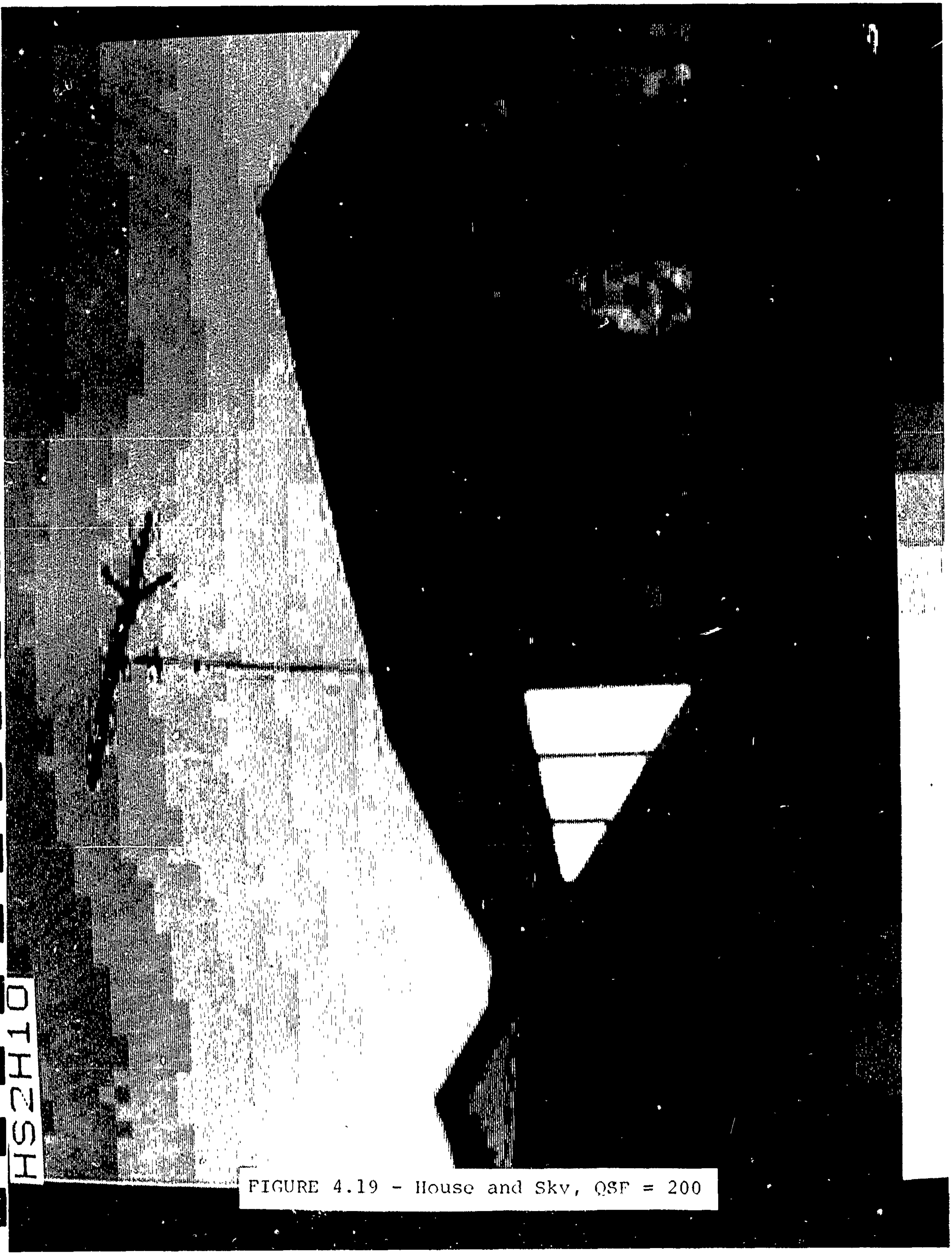


FIGURE 4.19 - House and Sky, QSF = 200

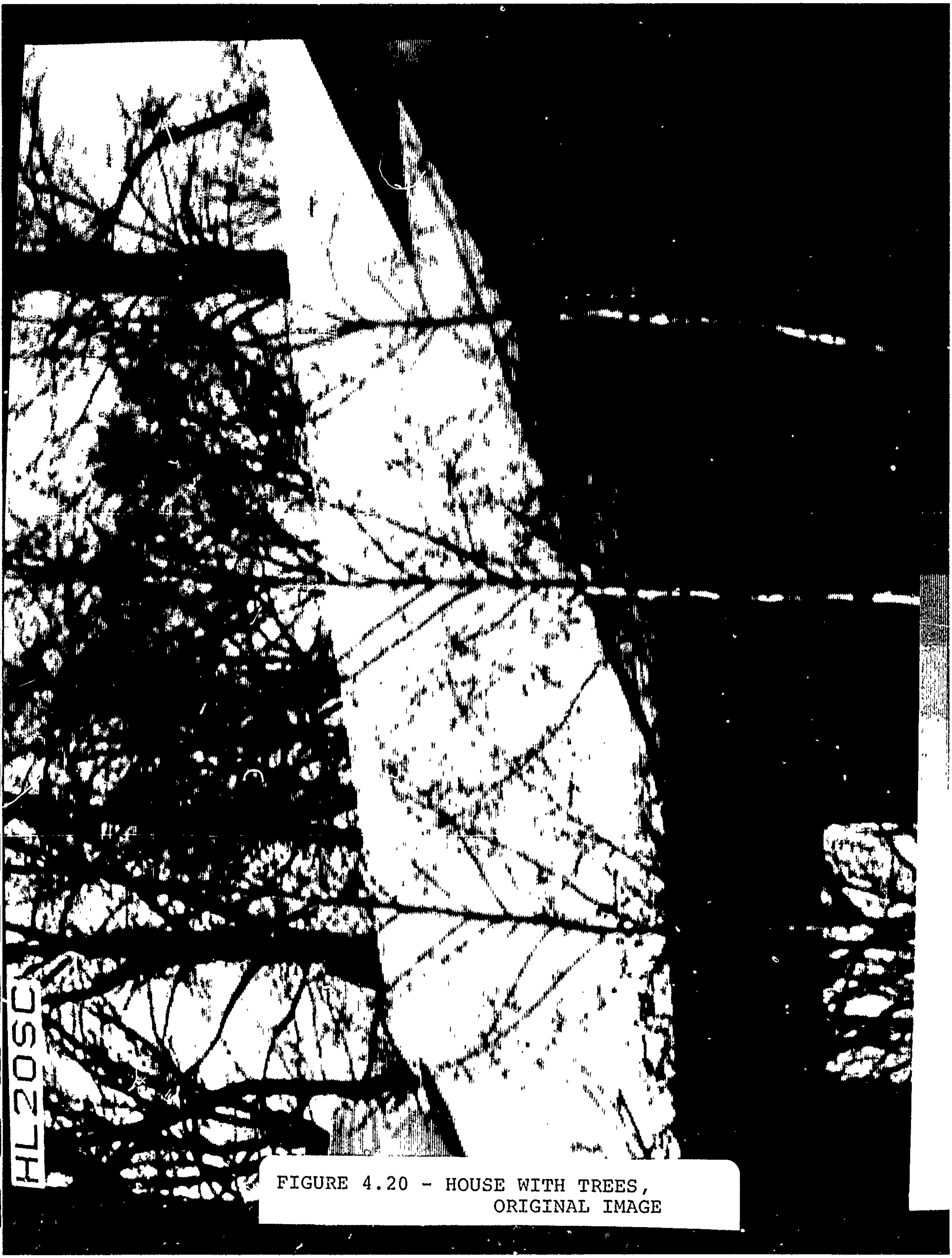


FIGURE 4.20 - HOUSE WITH TREES,
ORIGINAL IMAGE

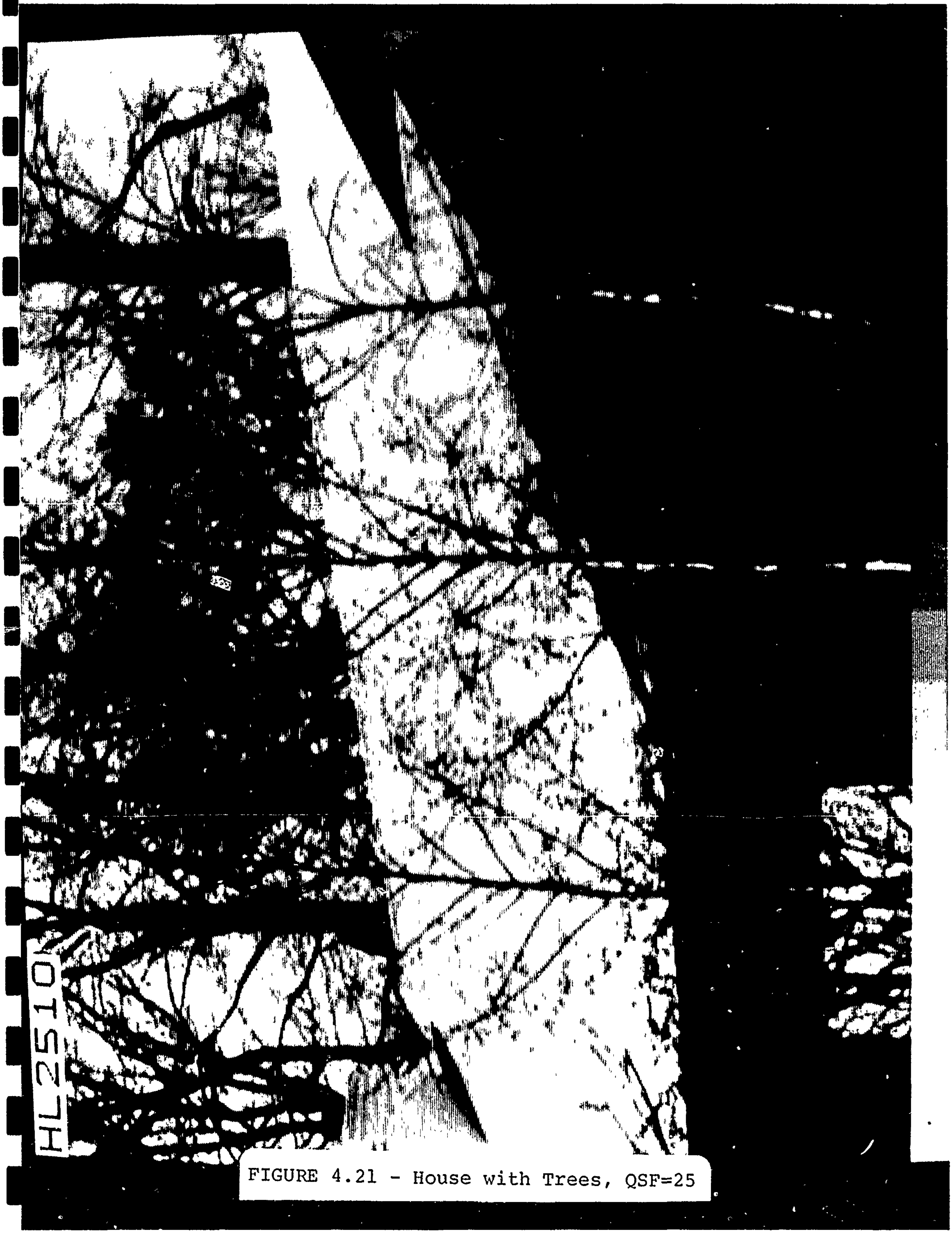


FIGURE 4.21 - House with Trees, QSF=25

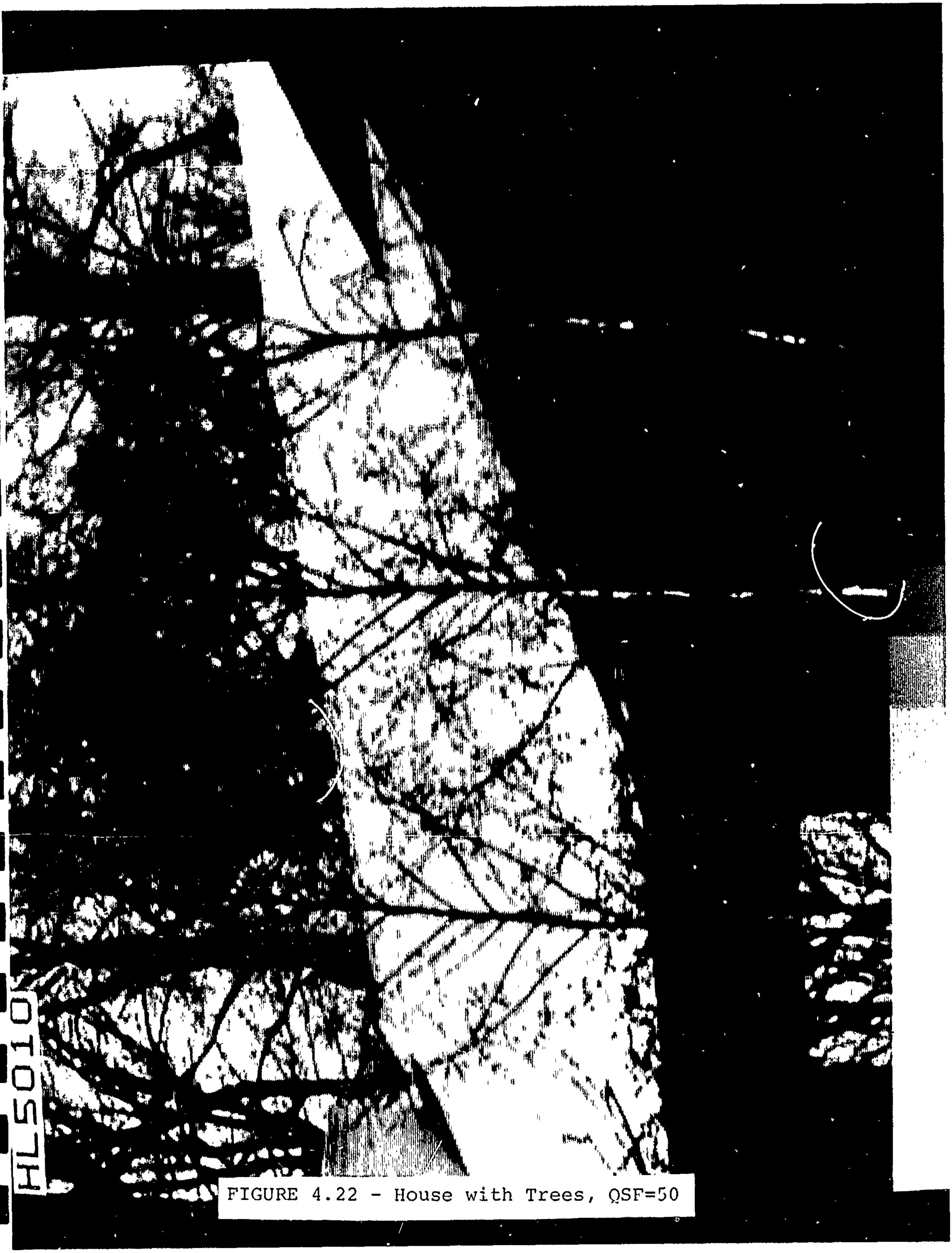


FIGURE 4.22 - House with Trees, QSF=50

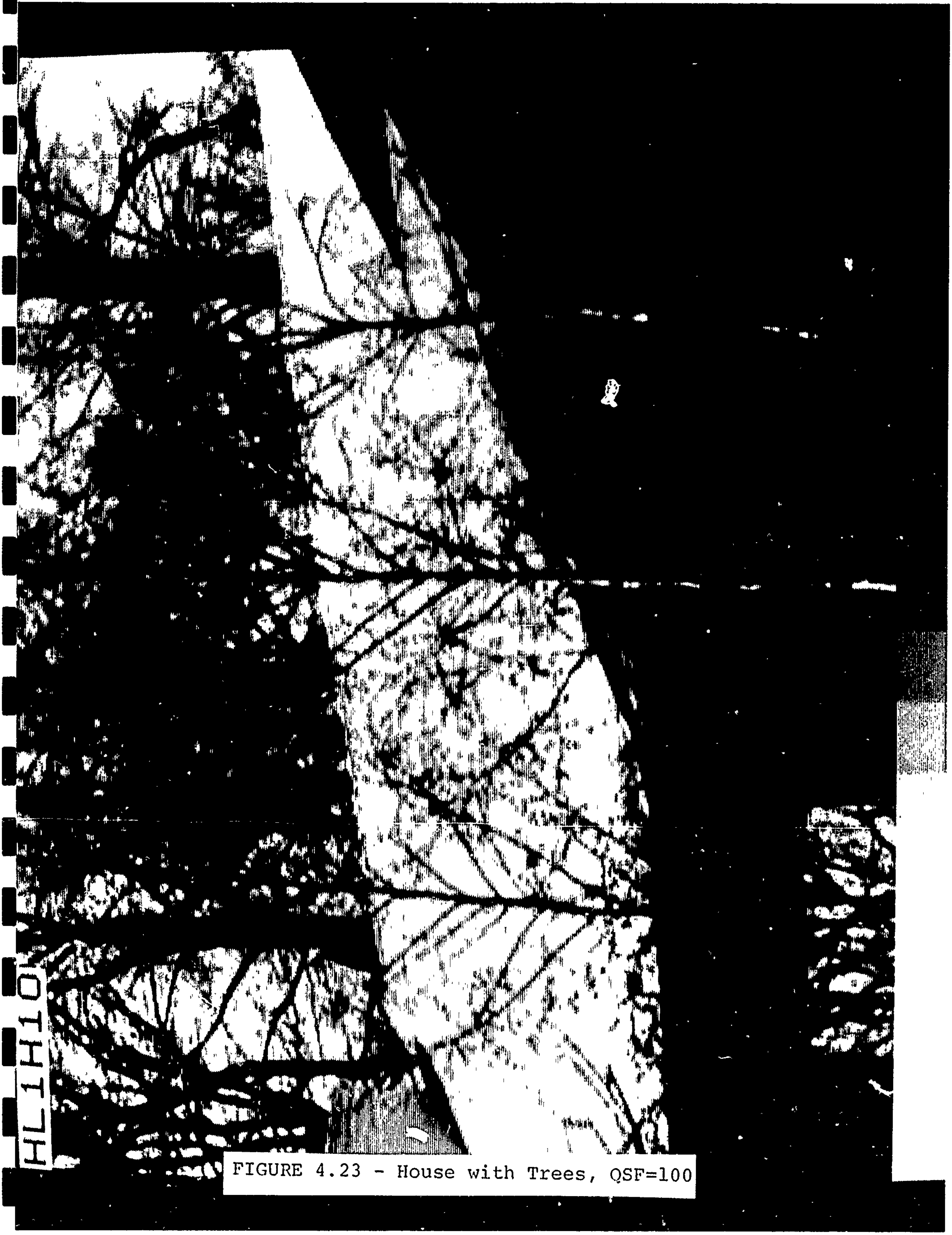


FIGURE 4.23 - House with Trees, QSF=100

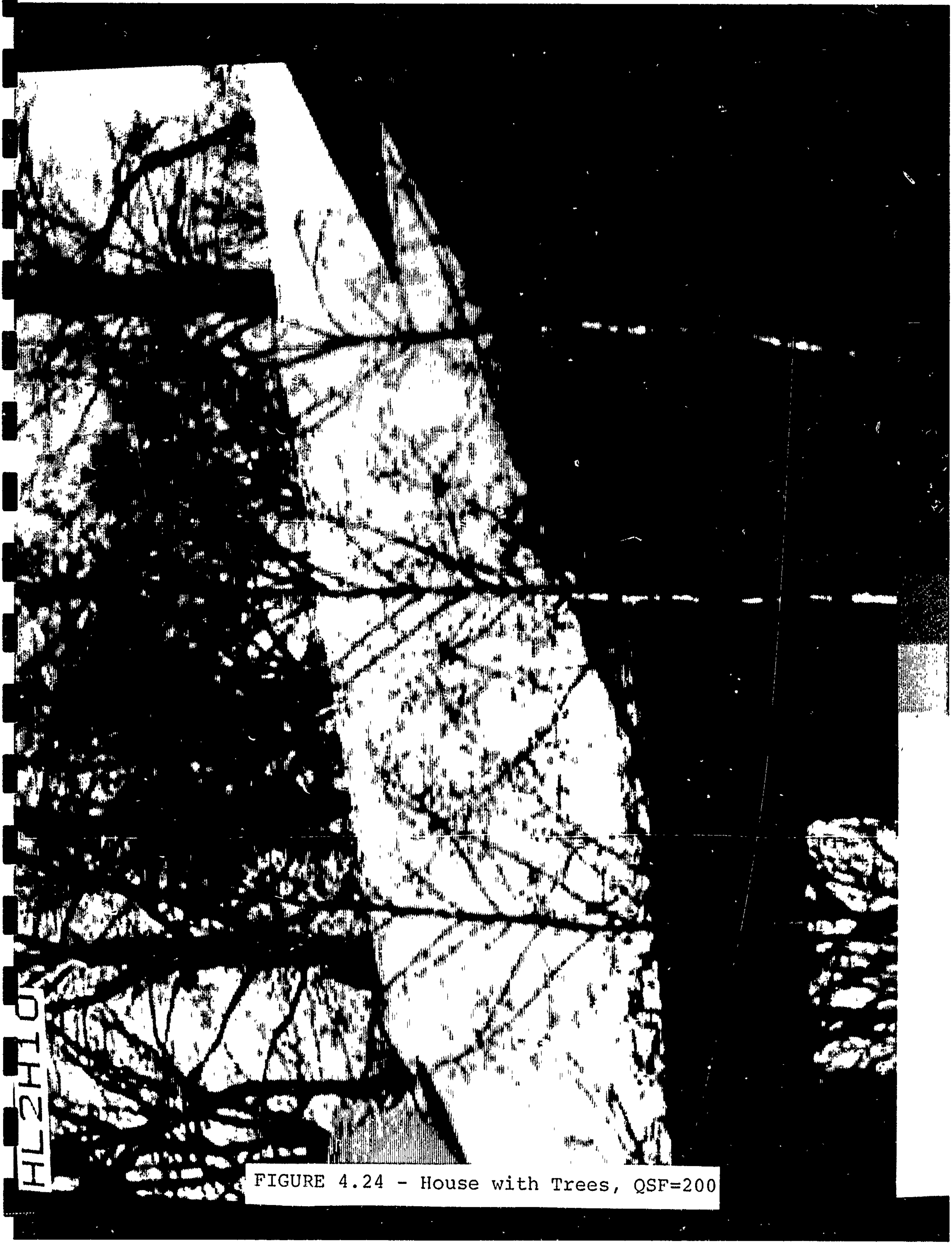


FIGURE 4.24 - House with Trees, QSF=200

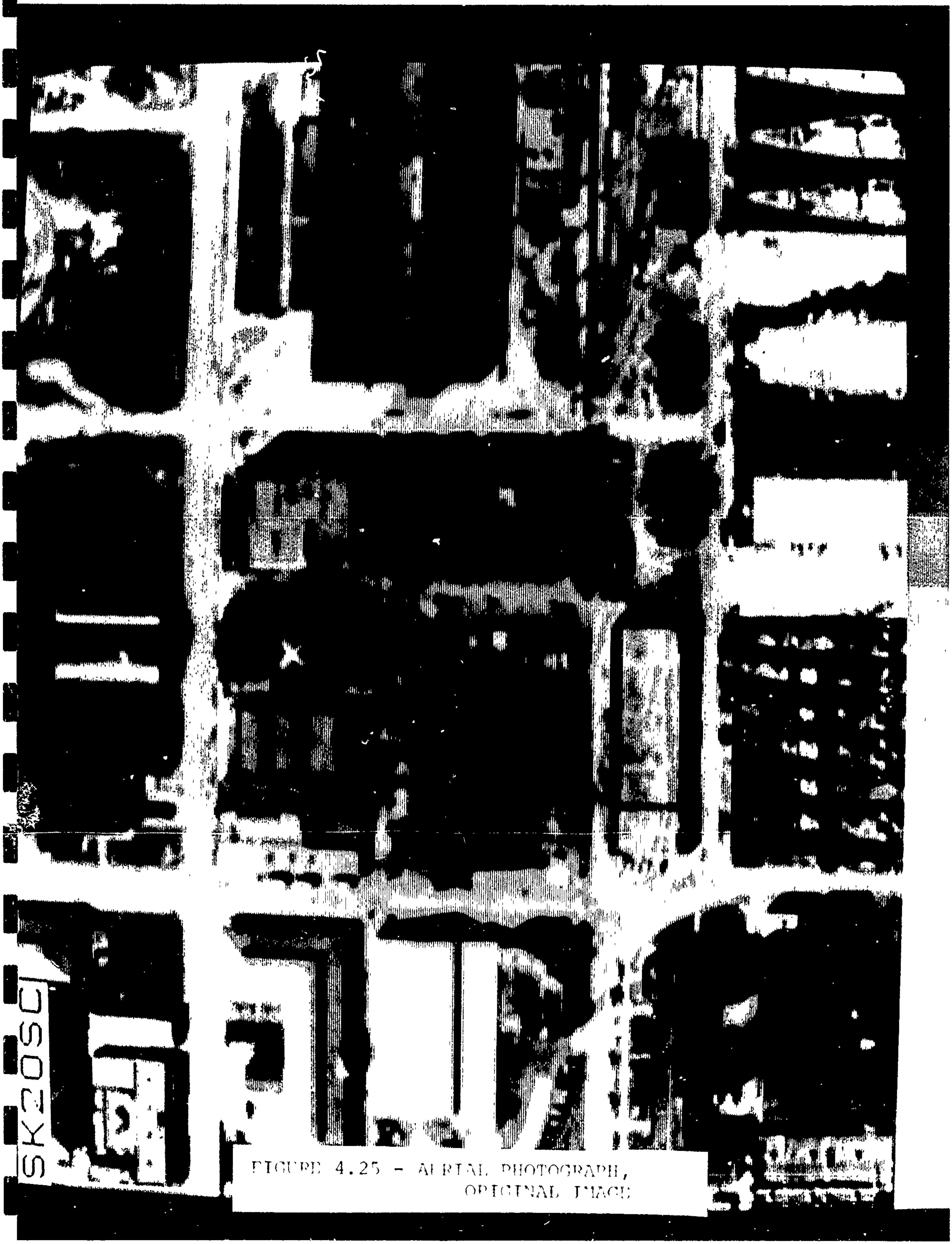


FIGURE 4.25 - AERIAL PHOTOGRAPH,
ORIGINAL IMAGE

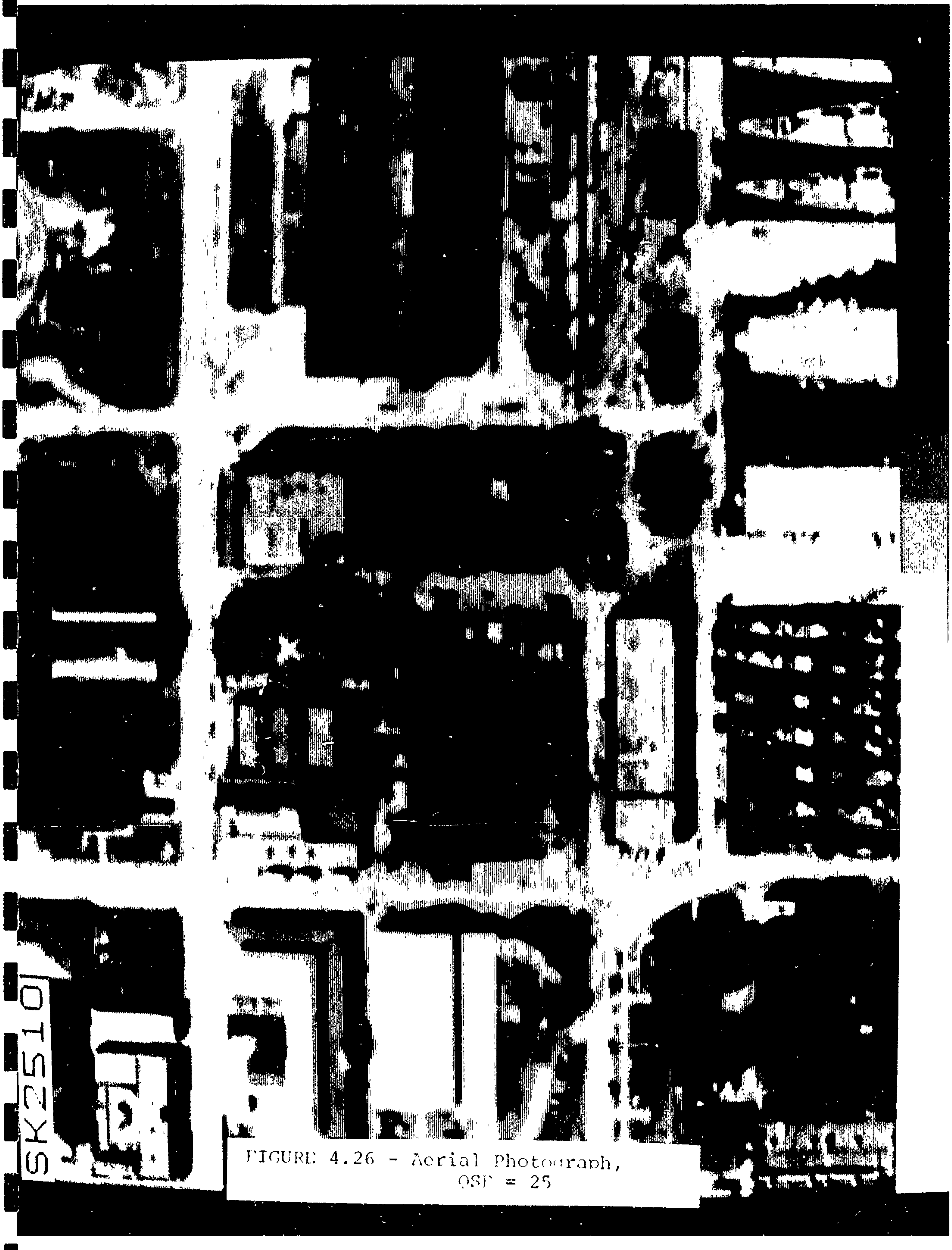


FIGURE 4.26 - Aerial Photograph, QSF = 25

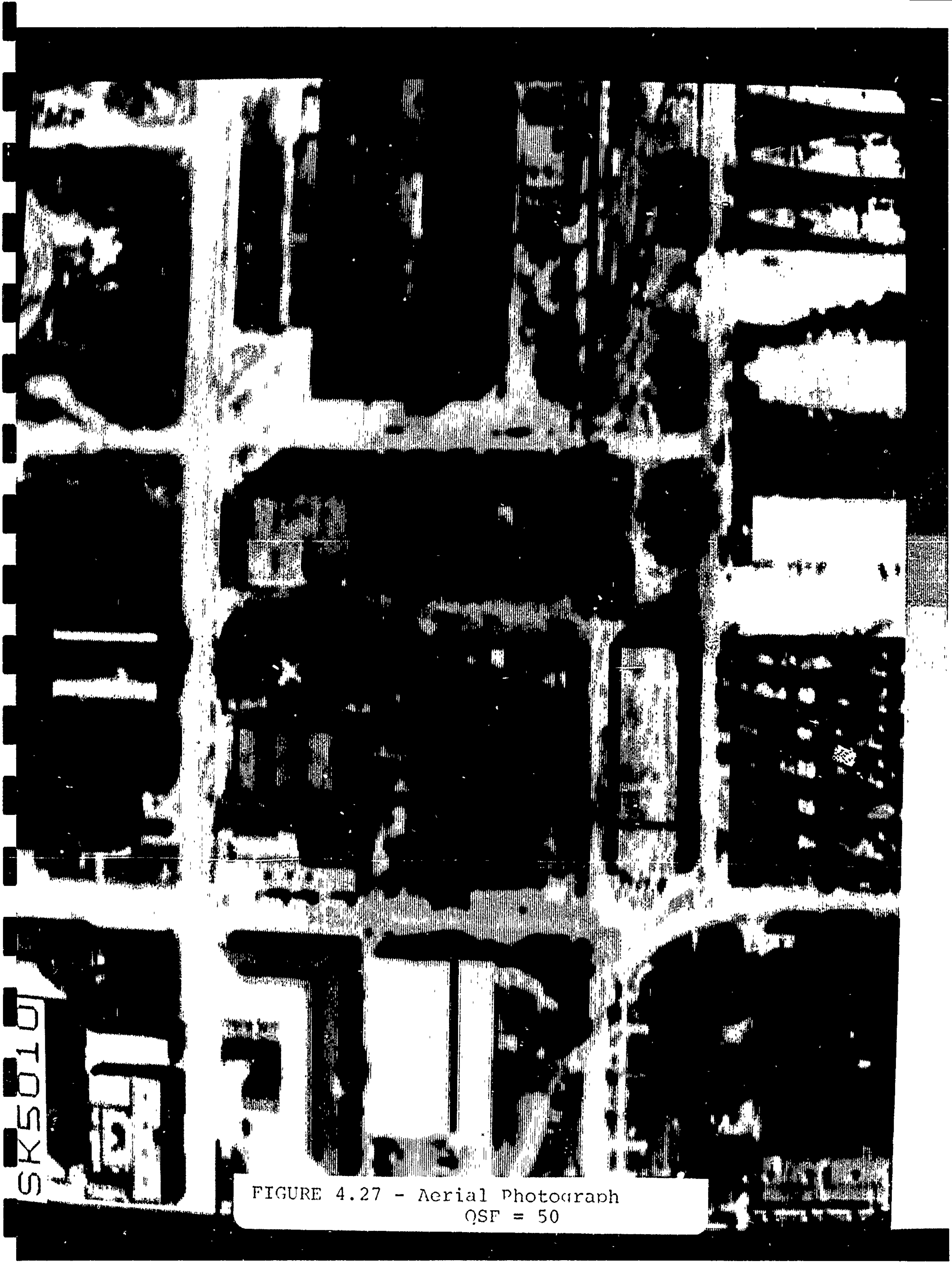


FIGURE 4.27 - Aerial Photograph
QSF = 50

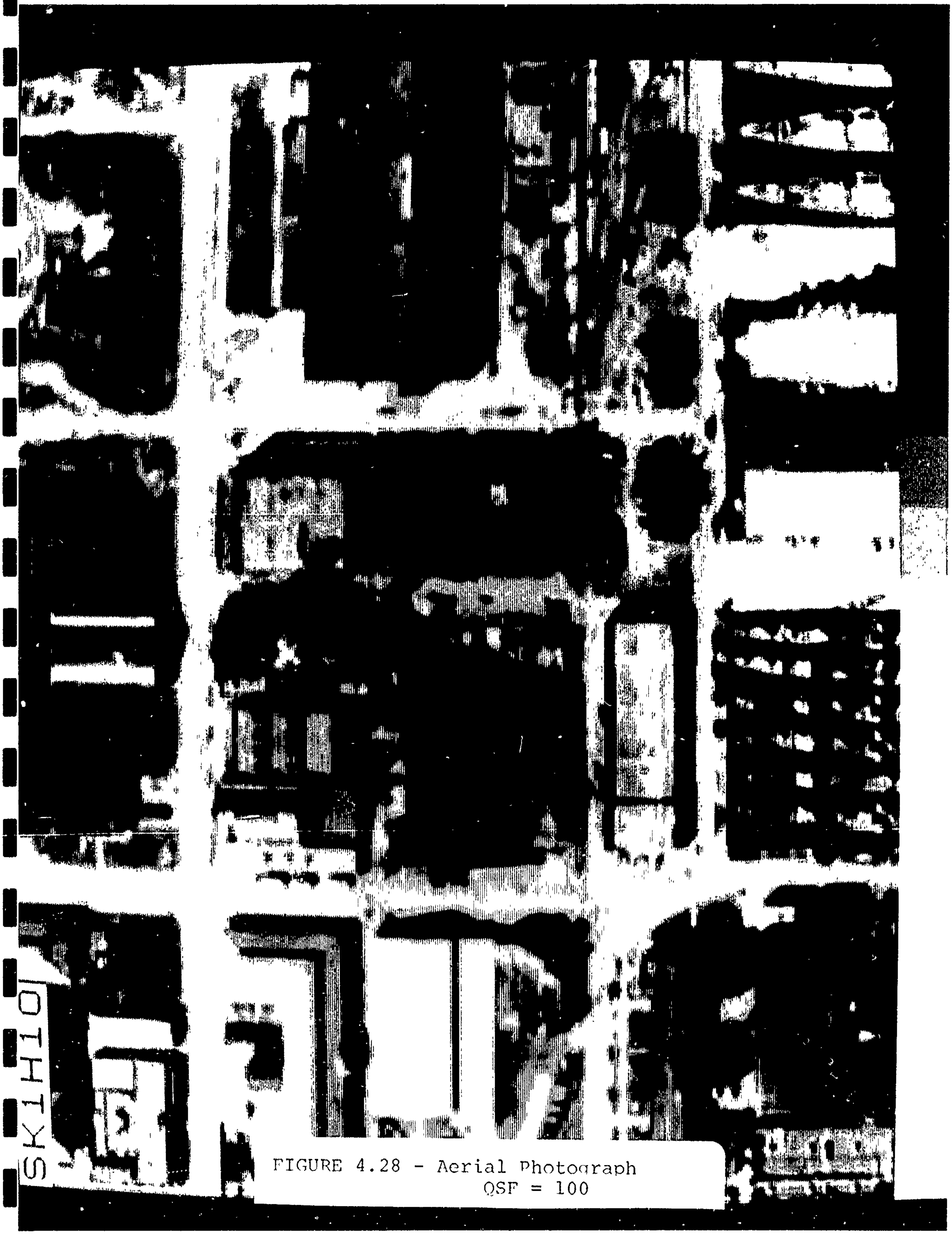


FIGURE 4.28 - Aerial Photograph
QSF = 100

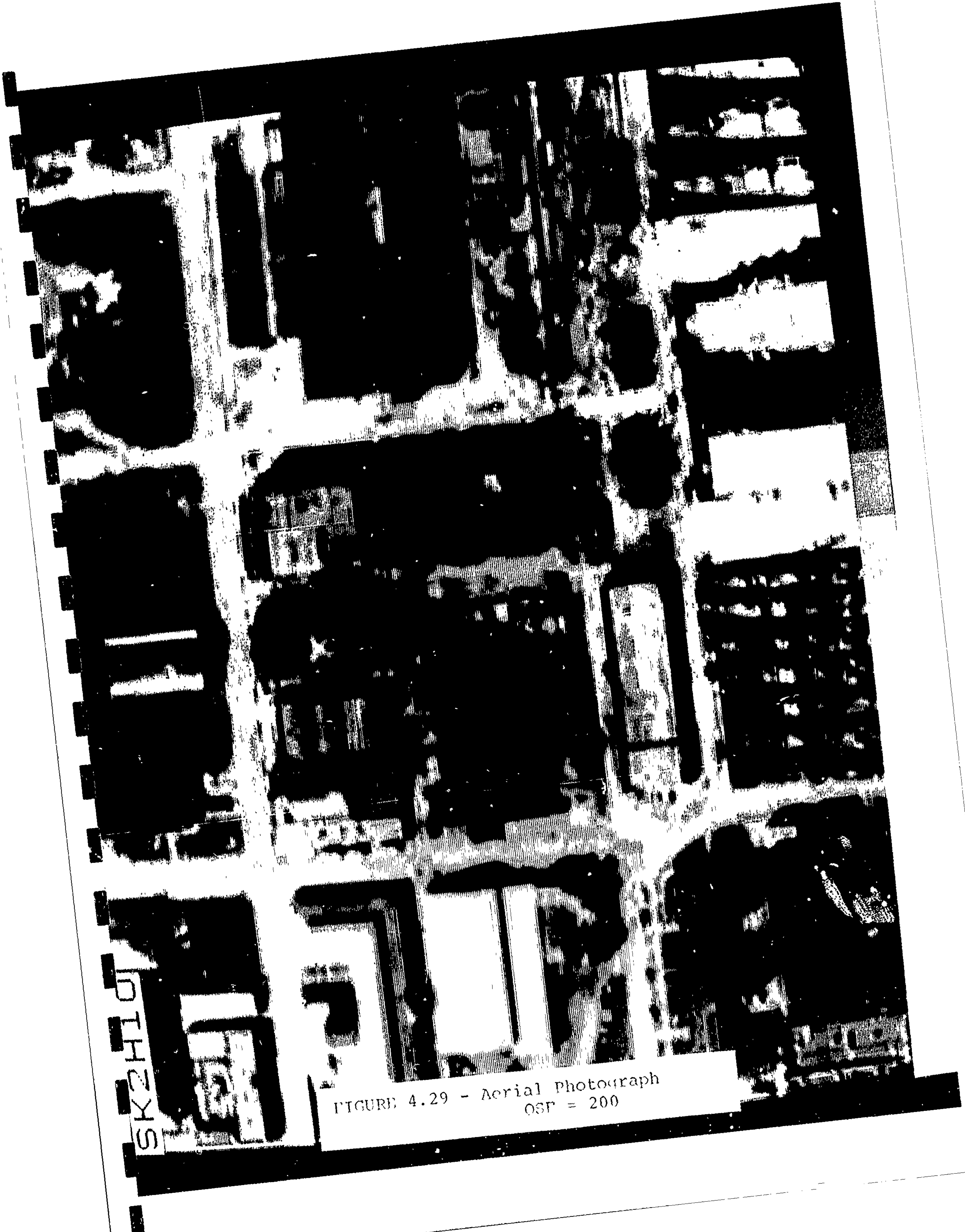


FIGURE 4.29 - Aerial Photograph
OSP = 200

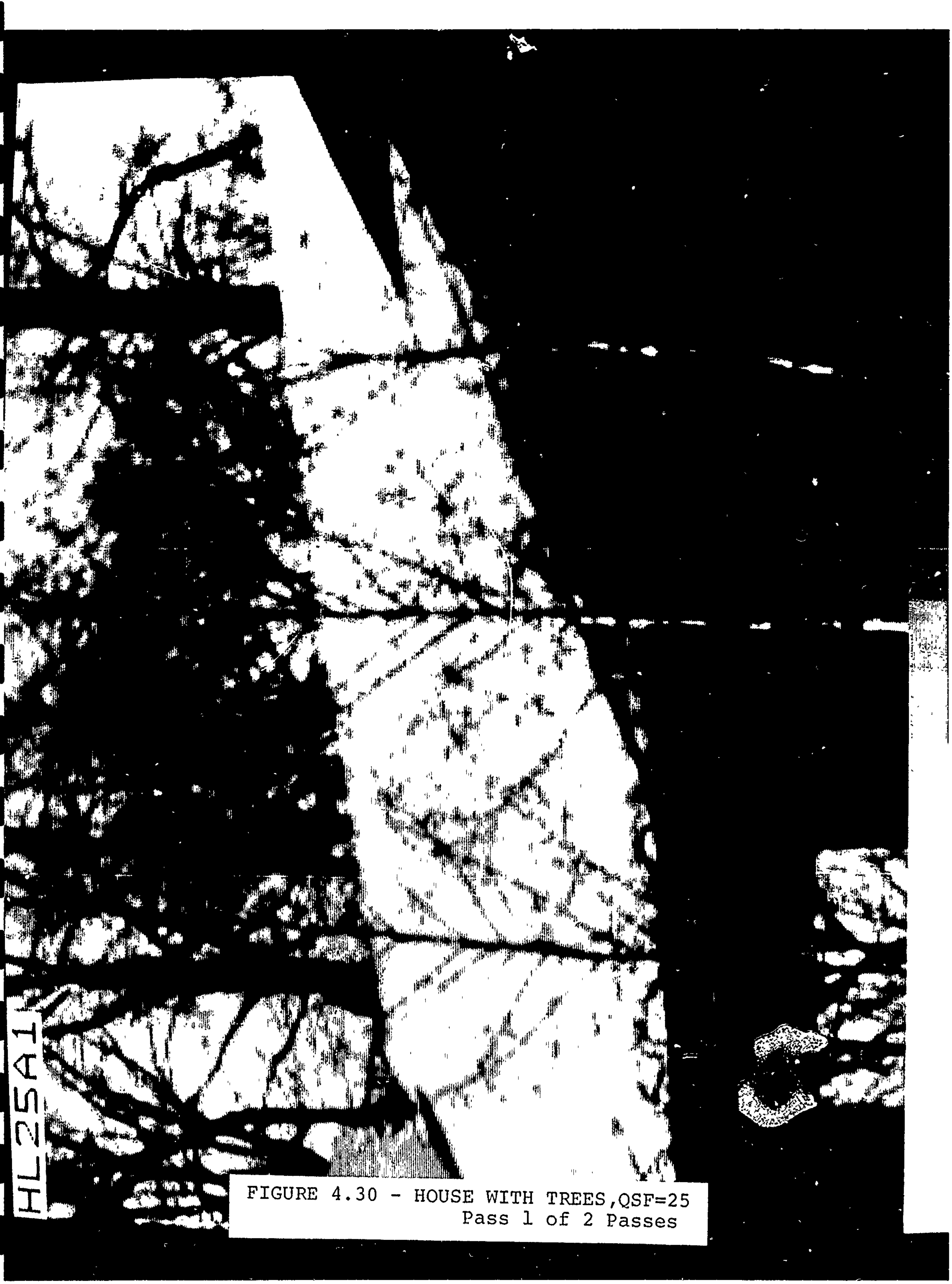


FIGURE 4.30 - HOUSE WITH TREES, QSF=25
Pass 1 of 2 Passes

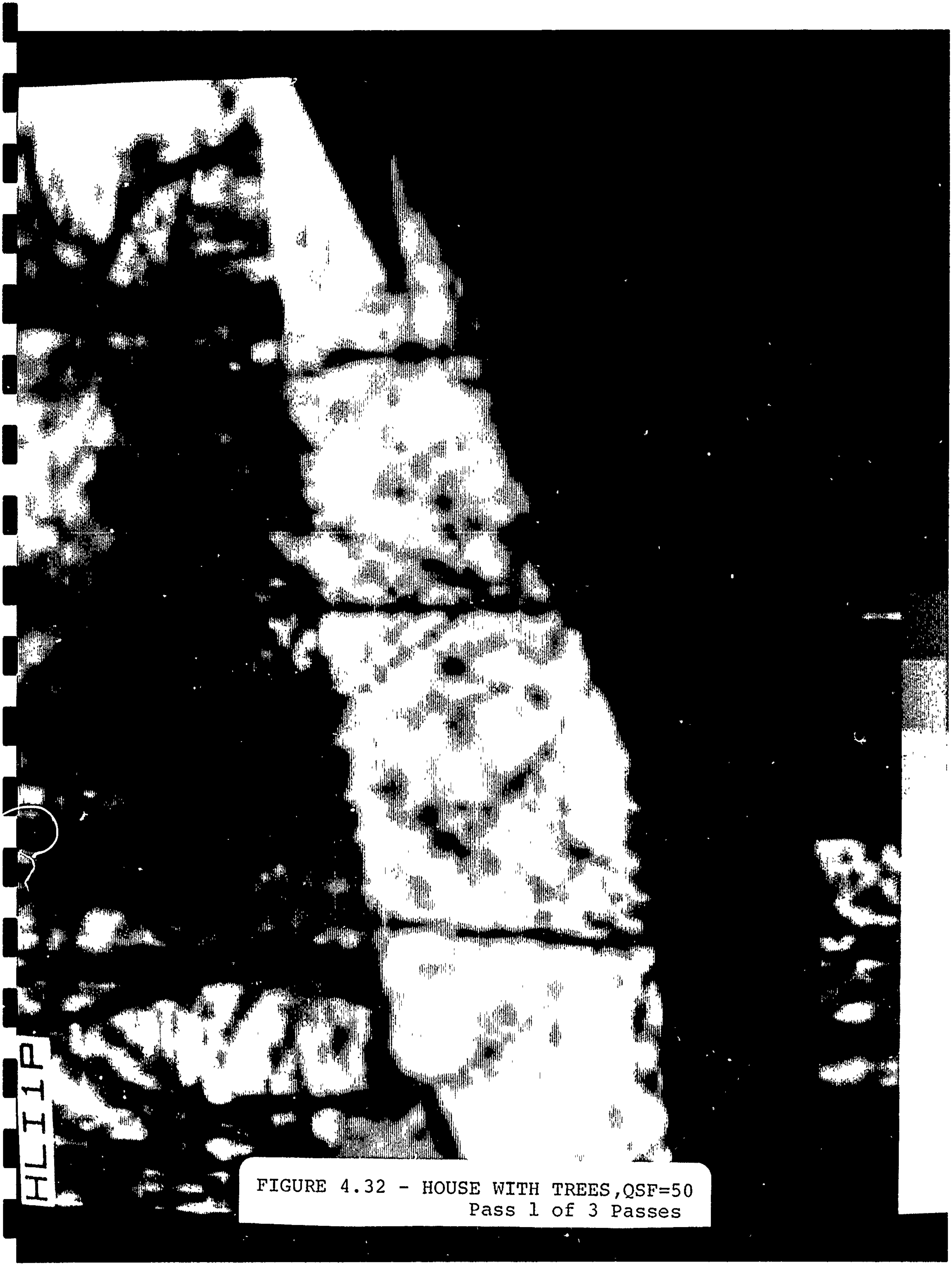


FIGURE 4.32 - HOUSE WITH TREES,QSF=50
Pass 1 of 3 Passes

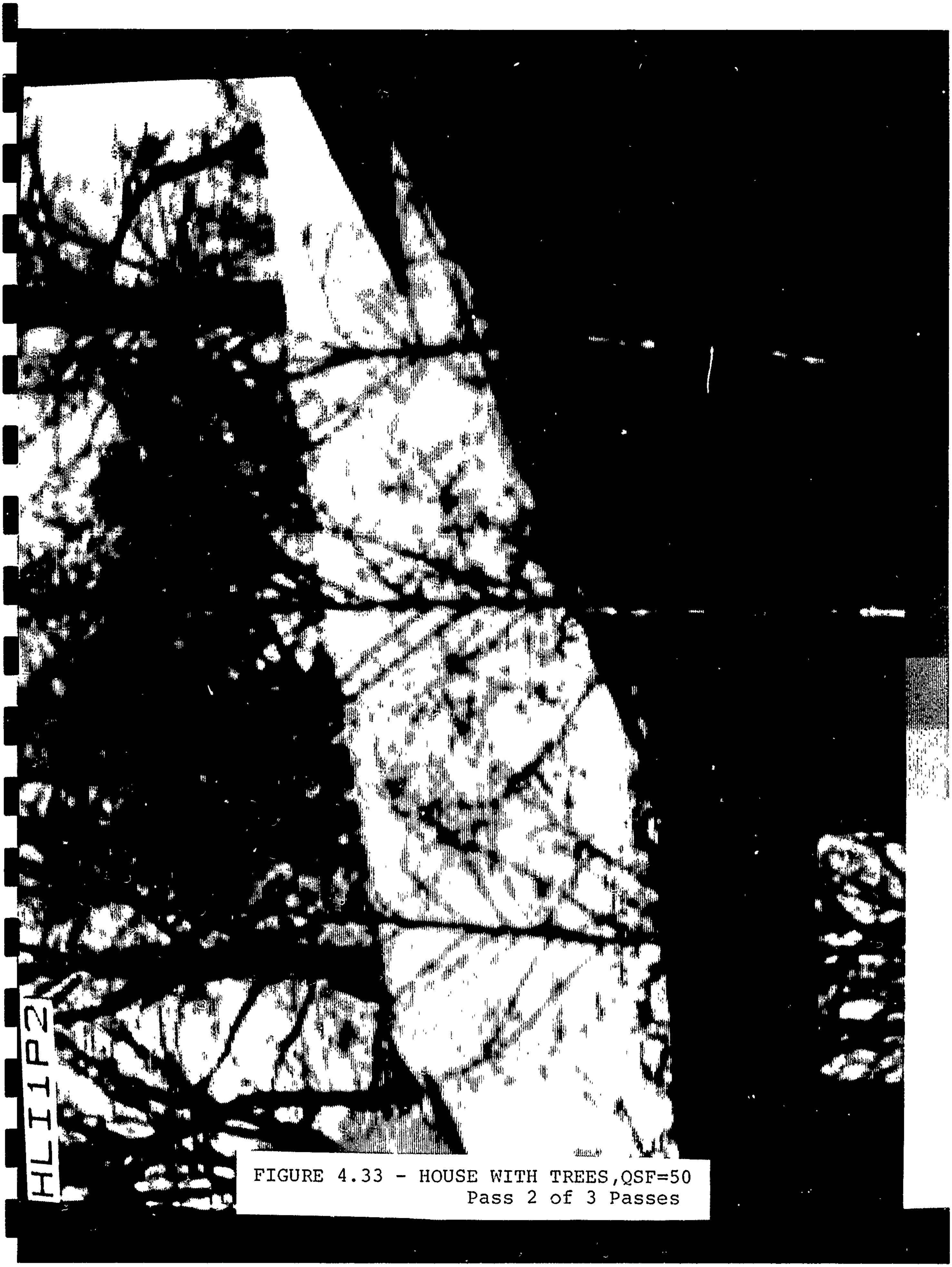


FIGURE 4.33 - HOUSE WITH TREES,QSF=50
Pass 2 of 3 Passes

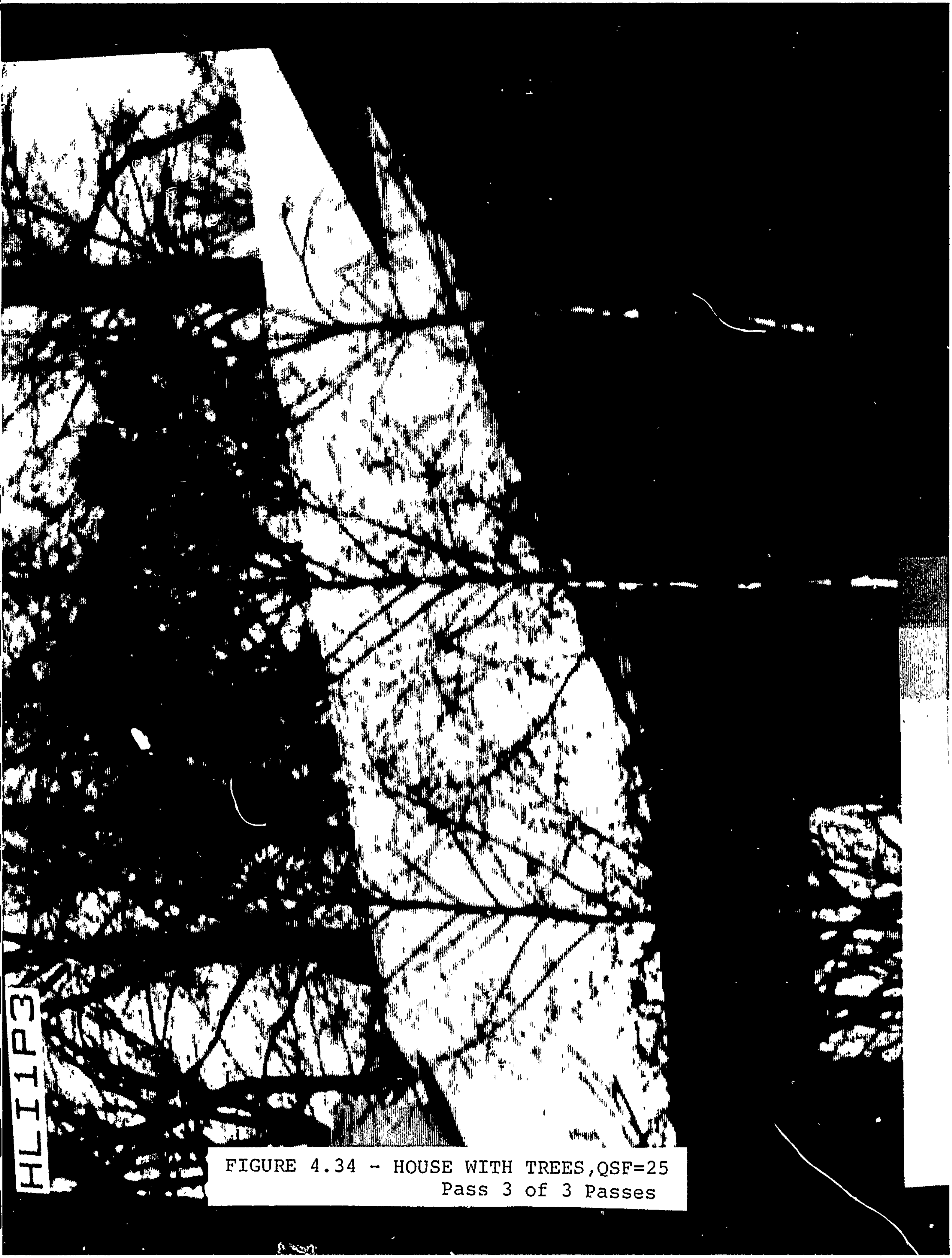


FIGURE 4.34 - HOUSE WITH TREES,QSF=25
Pass 3 of 3 Passes

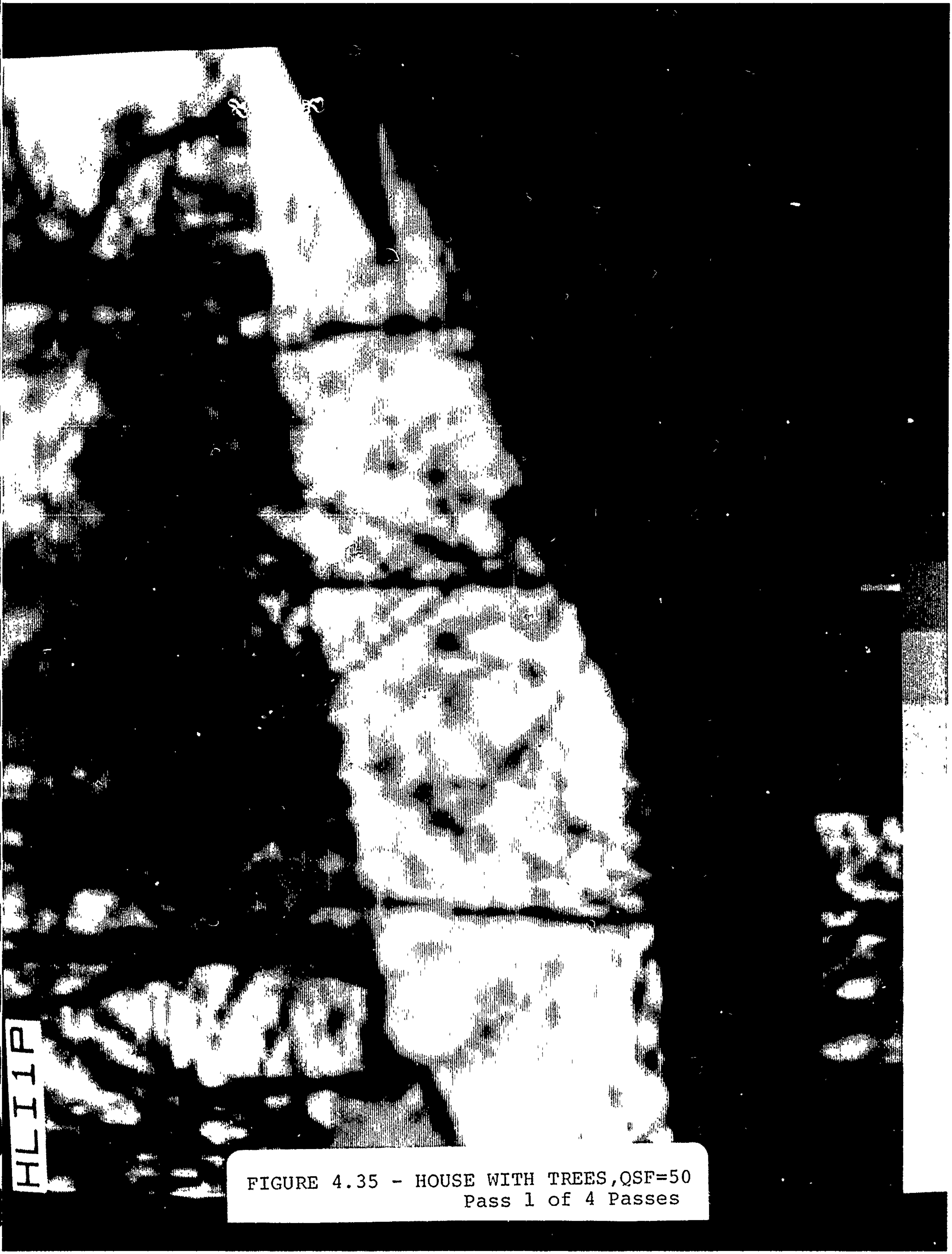


FIGURE 4.35 - HOUSE WITH TREES,QSF=50
Pass 1 of 4 Passes

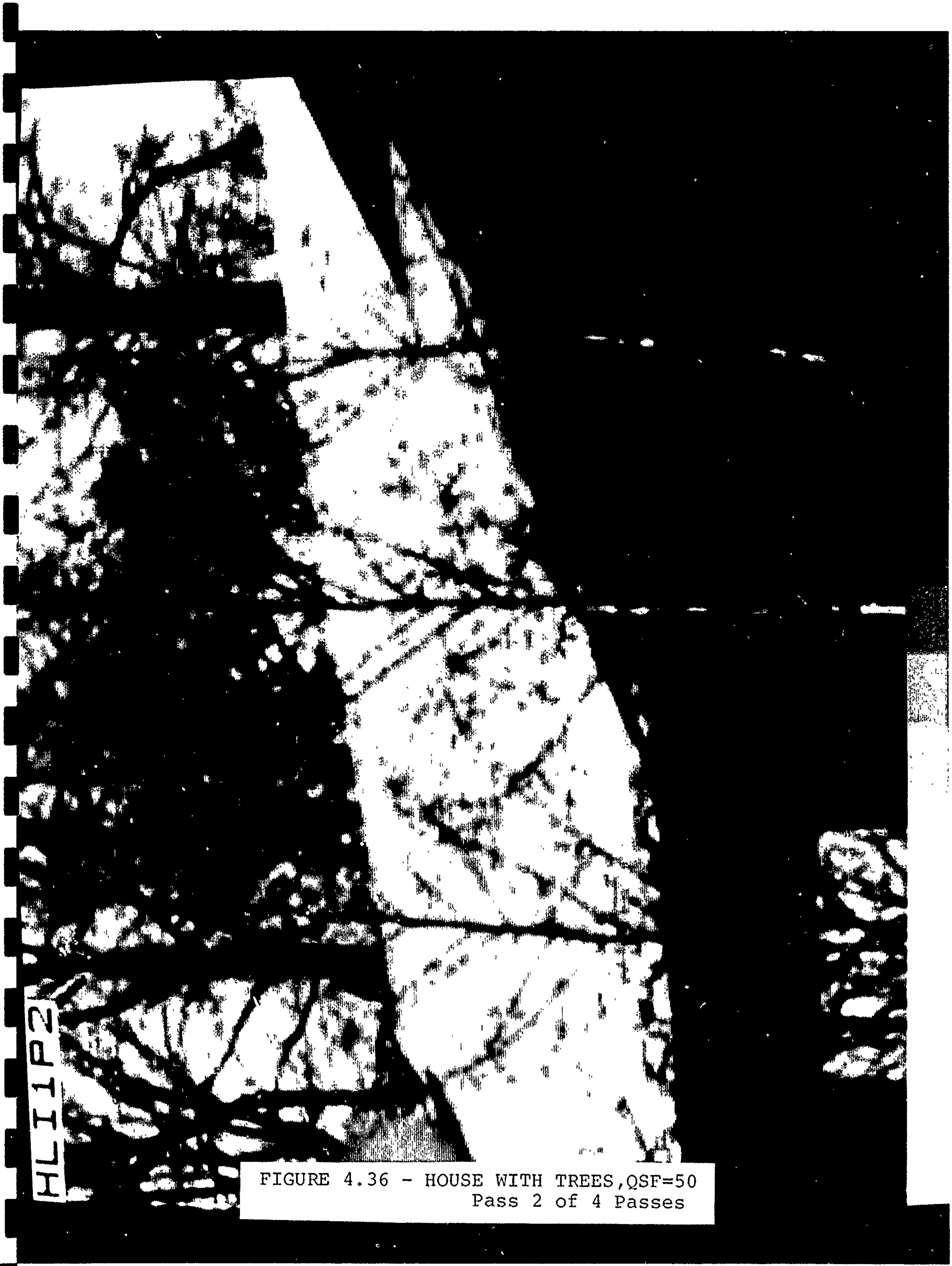


FIGURE 4.36 - HOUSE WITH TREES,QSF=50
Pass 2 of 4 Passes

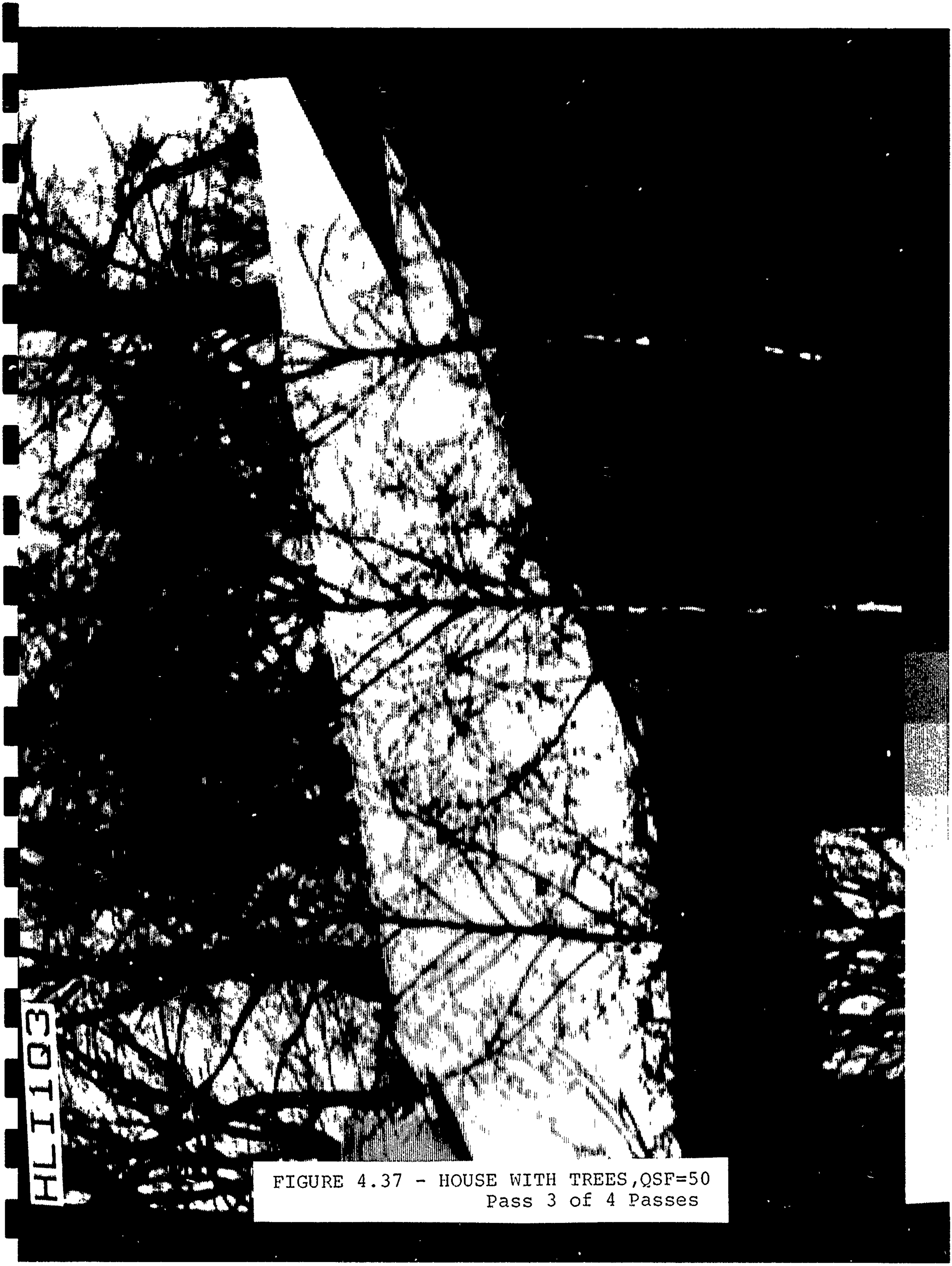


FIGURE 4.37 - HOUSE WITH TREES,QSF=50
Pass 3 of 4 Passes

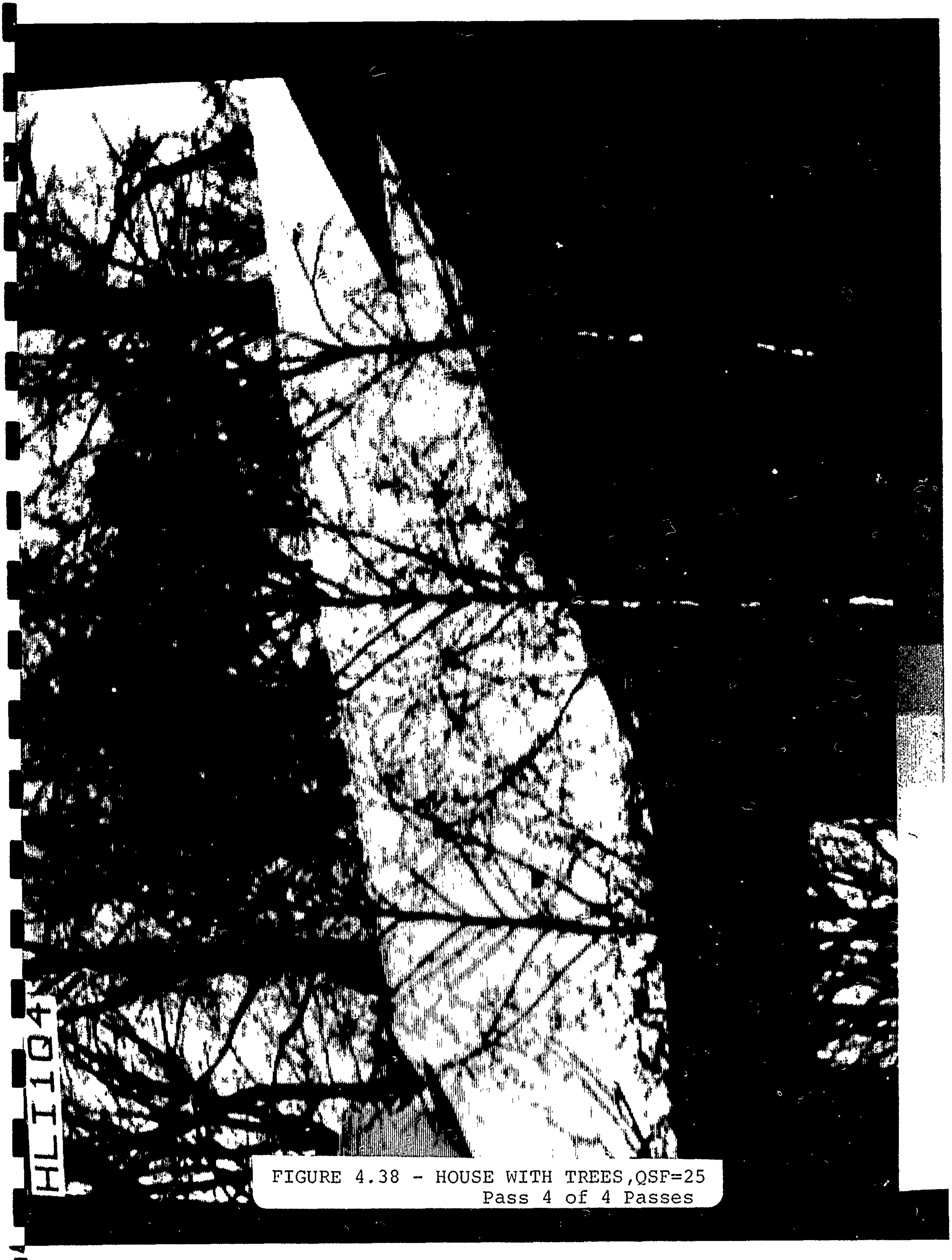


FIGURE 4.38 - HOUSE WITH TREES,QSF=25
Pass 4 of 4 Passes

5.0 Conclusions and Recommendations

5.1 Conclusions

The ADCT Simulation System was built around a sequential transmission simulator. Sequential transmission is the core of the baseline system standard proposed by JPEG. The transmitter performs the forward DCT, quantizes the DCT coefficients and losslessly encodes and transmits the quantum numbers. The receiver decodes and "dequantizes" the coefficients and performs the inverse transform. Quantization and "dequantization" are controlled by a "visibility" matrix (now, more appropriately, called a quantization matrix), which, together with a quantization scale factor, determine the quantum step size for each coefficient in an 8 by 8 block, and thereby the fidelity of the coded image.

The simulator combines the functions of both transmitter and receiver, since this is required of the transmitter in all progressive transmission methods except for transform-domain bit slicing.

The Q coder was employed instead of Huffman coding to permit a direct comparison of the study and JPEG results of transmitting JPEG images. The Simulation System, however, employs a simplified version of the Q Coder model, even though it was known that this would yield somewhat worse data compression than the JPEG model. The study model nevertheless provided a common basis for comparing various progressive transmission modes to sequential transmissions. The compression difference was sufficiently great, however, that the more complex JPEG model be employed if the Q coder is adopted for the standard.

Table 5.1 compares the various transmission modes studied. The comparison is one of data compression vs. system complexity; the final images produced by all transmission modes had almost identical RMS error values with respect to the original image. Appendix A is a discussion of some of the factors that control image fidelity and data compression in the various transmission modes.

Transmission Mode	Compression (Compared to Sequential Mode)	Computational Requirements	Major Storage Requirements
Sequential	-	DCT, IDCT (receiver only), quantization, "dequantization," coding, decoding	One 8-line buffer
Transform-Domain Bit Slicing	A few percent better than sequential	Same as sequential plus "history" processing and quantum number updates	Same as sequential plus quantum numbers for entire image
Image-Domain "Bit Slicing"	More than 30 percent worse than sequential	Same as sequential plus IDCT in transmitter plus image addition and subtraction	Same as sequential plus previous image refinement
Hierarchical Progression	A few percent worse than sequential	Same as image-domain "bit slicing" plus filtering, subsampling and interpolation	Same as mage-domain "bit slicing"

Note: IDCT = Inverse Discrete Cosine Transform

Table 5.1 Transmission Mode Comparison

It is emphasized that these results are based on the use of the Q Coder, and would not necessarily be valid with Huffman coding. Transform-domain bit slicing was not simulated, because JPEG literature covers this mode extensively. Image-domain "bit slicing," however, gives the same image fidelity, with considerably worse compression.

This study has led to the following conclusions and recommendations for a DCT-based Group 4 facsimile system:

- o A minimum system should employ the sequential transmission mode.
- o No recommendation is made for or against the Q Coder instead of Huffman coding. While the Q Coder is claimed to give 10 percent better compression, it is more complex that the Huffman Coder.
- o The JPEG baseline standard (with Huffman or Q coding) should be adopted. Color and resynchronization capability are optional; they are not required in Group 4 facsimile.
- o If the Q Coder is adopted, transform-domain (JPEG) bit slicing should be seriously considered, because it gives the best compression. However, it is also the most complex algorithm studied and requires a great deal of memory.
- o Predicting DC and AC coefficients enhances compression by a few percent at the expense of considerably more computational complexity.

If an interactive mode is desired, in which the receiving party views the incoming image in various stages of refinement and informs the transmitter when an acceptable image has been received, then hierarchical progression or transform-domain bit slicing should be employed as an optional extension, even though full refinement to high image quality gives slightly worse compression in the hierarchical case than a sequential transmission of equal quality. As explained in Appendix A, a hierarchical progression should never refine an image more than once at any given level of sub-sampling and interpolation, else data compression suffers. This means that the fourth stage of the progression shown early in Doc. N800 should be omitted, and a lower quantization scale factor should be employed for the third stage to yield the desired final image quality.

Image-domain "bit-slicing" is simpler than hierarchical progression, but much less efficient. It should be substituted only if system simplicity is more important than data compression.

DIS results show that the ending images of progressive and sequential transmissions are almost identical, provided the quantization scale factor in the last pass of the former is the same as that in the latter. This observation is discussed in detail in Appendix A. In transform-domain bit slicing the images are exactly identical.

5.2 Recommendations for Further Study

The results reported herein are based on the use of the Q Coder. Doc. N800, and JPEG proposals for the baseline system, specify Huffman coding. Huffman coding requires coding models that are completely different from Q Coder models. Huffman coding depends on the frequencies with which members of a large set of symbols are coded, and the most frequently coded symbol requires at least one bit. With the Q Coder, symbol encoding consists of sequences of binary decisions, but frequently-used symbols can, in principle, be encoded in less than one bit.

Even though, as reported earlier, the Q Coder provides about 10 percent better compression than Huffman coding, the Q Coder is more complex. Moreover, the proposed standard for the baseline system specifies Huffman coding.

It is recommended that funds be provided to perform the following tasks:

- o Repeat the simulations performed under the current task with Huffman coding substituted for the Q Coder;
- o Simulate transform-domain bit slicing with Huffman coding to determine whether the improved compression with respect to sequential transmission still holds;
- o Test the conjecture that transform-domain bit slicing with the Q Coder is as effective, in terms of compression, in a sequential as in a progressive transmission. If so, then bit slicing can be employed without the very large amount of memory currently required.

No image photography or subjective evaluation would be required, since the proposed simulations would yield the same coded images as did the simulations reported herein.

Appendix A

Image Fidelity and Data Compression in Sequential and Progressive DCT Transmissions

A.1 Image Fidelity

In comparing various DCT-based transmission modes, an interesting question arises: When does a progressive build-up yield a final image identical to that produced by a sequential transmission? The following discussion addresses this question and proves some statements made in the body of this report.

Because the DCT is a linear transform, the unquantized DCT of the difference between two images is the difference between the DCT's of the two images taken separately. Therefore, image refinements can be analyzed in terms of the transform coefficients even when the refinements are accomplished by transforming difference "images."

This discussion is confined to the case wherein the final refinement of a progressive build-up is transmitted with the same quantization scale factor as the single sequential transmission. Were this not the case, the two results would be different, because the final quantized coefficients would be different.

The symbol S is used for the quantization scale factor, q, sometimes with subscripts, stands for the quantum step size of a given coefficient, and r, also with subscripts, denotes a coefficient quantum number.

A.1.1 JPEG Bit Slicing

The JPEG bit slicing method always produces the same coded image as does the sequential method. In the first pass, the transmitter computes and saves the quantum numbers for S = 25 and transmits these quantum numbers divided by 8. In succeeding passes, the transmitter transmits one-bit corrections, the receiver ultimately acquiring exactly the quantum numbers for S = 25. Since these quantum numbers are the same as those for a sequential transmission with S = 25, the inverse DCT's produce identical images in both cases. (This, of course, assumes that the same inverse DCT algorithm and precision are employed.)

A.1.2 "Bit Slicing" in the Image Domain

Image domain "bit slicing" is similar to JPEG bit slicing in that the transmissions are made with S = 200, 100, 50 and 25, i.e., 8, 4, 2 and 1 times 25. The difference, as far as image fidelity is concerned, is that *difference* images are transformed and quantized in *each* refinement pass. It will now be shown that the currently-specified quantization rounding rules (equivalent to rounding the magnitude of a real division to the nearest integer) *preclude* a guarantee that this transmission mode produces the same image as a single sequential transmission with S = 25.

Let I be the original image and C be any *unquantized* DCT coefficient of that image. Let q_n be the quantization step size for this coefficient in transmission pass n. Then q_{n+1}, the step size in pass n+1, is $q_n/2$. Let I_n be the approximate image resulting from transmission pass n.

In the first pass, Image I is transformed, and coefficient C is quantized to step size q_1, giving quantum number $r_1 = C/q_1$ rounded to the nearest integer, and quantized coefficient $C_1 = r_1 q_1$. The inverse DCT of the quantized coefficients gives the first image approximation, I_1.

In the second pass, I_1 is subtracted from I, and the difference image is transformed. Let D be the unquantized coefficient of the difference image corresponding to the original C. Because the DCT is linear, $D = C-C_1 = C-r_1 q_1$. Because C was quantized to the nearest quantum step in the first pass, the absolute value of D is less than or equal to $q_1/2$, always less if q_1 is odd. In general, for pass n+1, D is less than (or equal to) $q_n/2$. Since q_{n+1} is $q_n/2$ (truncated to an integer when q_n is odd), the absolute value of D is less than or equal to q_{n+1}. Therefore, when D is quantized to step size q_{n+1}, the quantum number, r_{n+1}, is always 0 or plus or minus 1. It is 1 when $D \geq q_{n+1}/2$; 0 when $-q_{n+1}/2 < D < q_{n+1}/2$; and -1 when $D \leq -q_{n+1}/2$.

Consider the special case wherein, in the next-to-last refinement, $C_n = kq_n$ and C (the original, unquantized coefficient) is $C_n - q_{n+1}/2$, where k is a positive integer. (A similar treatment applies, with sign changes, when k is negative.) The DCT of the difference image for the final refinement gives unquantized difference coefficient $D = C-C_n =$

$-q_{n+1}/2$. According to the rounding rules for negative numbers, $-q_{n+1}/2$ rounds to a quantized difference of $-q_{n+1}$, not 0. Adding the inverse-transformed difference image to the approximate image from the previous refinement is equivalent, in the transform domain, to adding the quantized difference, $-q_{n+1}$, to C_n, giving $C_{n+1} = C_n - q_{n+1}$. Thus, in the final refinement, C is in effect rounded <u>down</u> by $q_{n+1}/2$. In the sequential case, since $C = kq_n - q_{n+1}/2 > 0$ for $k > 0$, the quantized C is rounded <u>up</u> by $q_{n+1}/2$ to $kq_n = C_n$. Thus, in this special case, C is rounded <u>up</u> in the sequential case and <u>down</u> in the "bit-slicing" case, giving a difference of q_{n+1}.

The special case can be shown (mathematically or by exhaustive test) to occur when the final step size, q_4, is even and $|C| = [p + (m/16)]q_1$, where p is any non-negative integer, m = 3, 7, 11 or 15, and q_1 is the first (largest) step size; $q_1 = 8q_4$. Since the unscaled "visibility" matrix specifies quantum step size $q_3 = 2q_4$ for each coefficient, the special case can occur only for visibility matrix elements which are multiples of 4. For example, the DC coefficient visibility matrix element value is 16. Therefore, $q_1 = 64$, $q_2 = 32$, $q_3 = 16$ and $q_4 = 8$. Since, for q_1, there are 4 values of C for each p that fit the special case (e.g., 12, 28, 44 or 60 for p = 0), there is roughly one chance in 16 that the special case can occur for this coefficient. (The distribution of the unquantized DC coefficient across the quantum steps determines the actual probability.) Thus, image domain "bit slicing" can produce a slightly different image from that produced by a single sequential transmission.

When |D| is different from $q_{n+1}/2$ ($q_4/2$), the quantized difference always rounds in the "correct" direction to make the final quantized coefficient the same as in the sequential case. The final difference is rounded in the "wrong" direction only in the special case, making the final quantized coefficients different by a whole final quantum step size. If the rounding rules were modified to round up algebraically instead of in absolute value (e.g. -2.5 rounds up algebraically to -2), then the images produced by image domain "bit slicing" and sequential transmission would be identical.

A.1.3 Hierarchical Progression

In a hierarchical progression, the difference "image" transmitted during refinement is usually the difference between some image and the sub-sampled, coded and interpolated version of that image. Since interpolation takes place in the image domain, the unquantized DCT coefficients of the interpolated image are not necessarily on the quantum step boundaries determined when the sub-sampled image was quantized. Therefore, when the quantized difference image transform coefficients are in effect added to the unquantized coefficients of the interpolated image, a coefficient of the refined image is constrained to $j+kq_{n+1}$, with j not necessarily 0 or a multiple of q_{n+1}. Thus, the final quantized coefficient is not necessarily a multiple of q_{n+1}. In a sequential transmission, the quantized coefficient is always a multiple of the step size. Hence, the final refinement of a hierarchical progression is not necessarily the same as the result of a sequential transmission, even if the final step size is the same as for the sequential case.

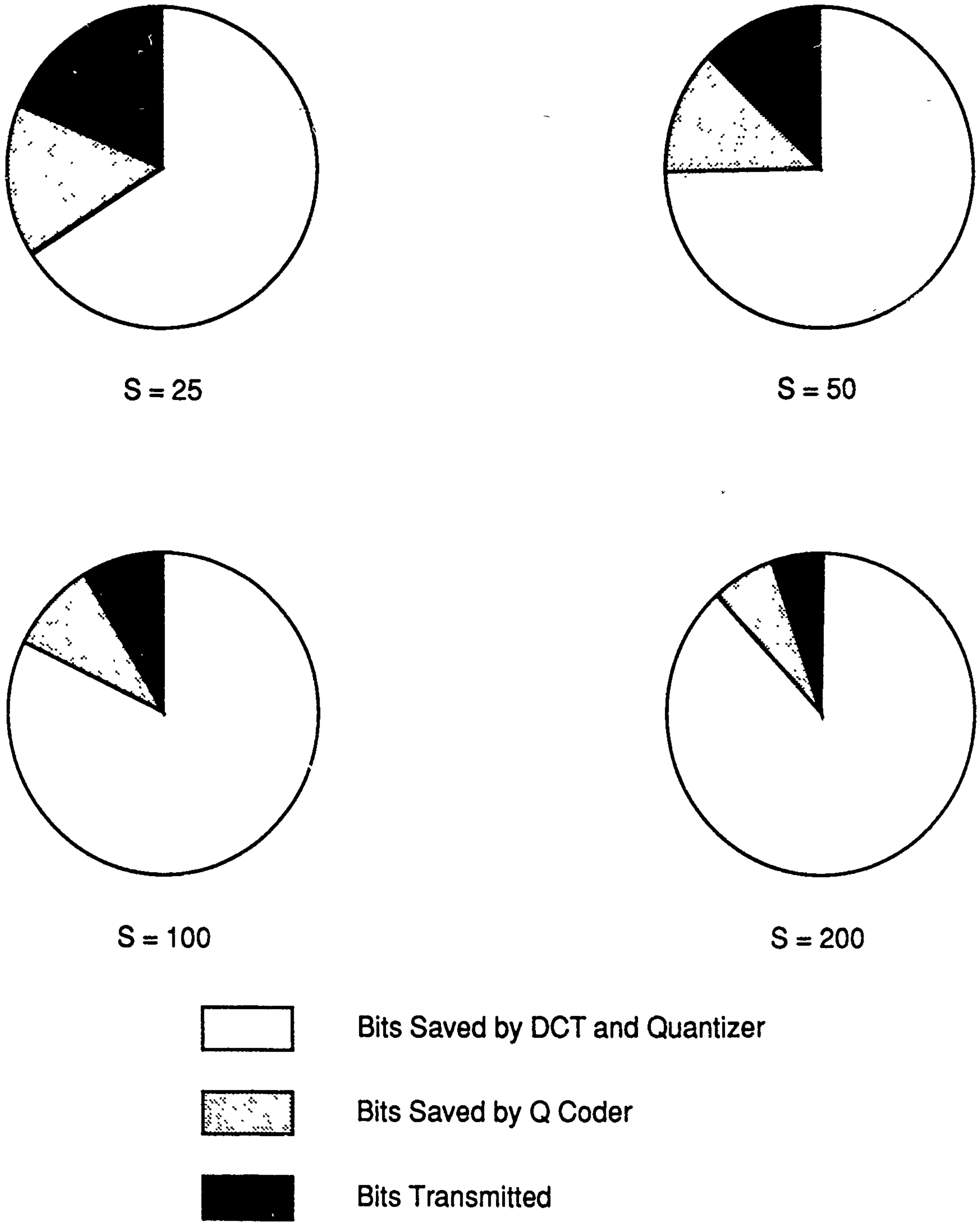
A.2 Data Compression

Since data compression comes partly from transforming the image and quantizing the coefficients and partly from entropy coding the quantum numbers, the question arises: How much does each contribute to the total compression? The answer is important, because, if the compression contributed by entropy coding were very small, then efforts to improve entropy coding techniques would yield little improvement in the total compression. Investigations showed that, although the DCT and quantizer do provide most of the data compression, the Q Coder roughly halves the number of bits that would be transmitted were entropy coding not employed.

It was conjectured that, even with the Q Coder, the number of bits required to transmit an image is, to good approximation, directly proportional to the number of non-zero quantum numbers. This conjecture is supported by study measurements and by JPEG investigations.[19]

To answer the question and test the conjecture, Program QECOEF was instrumented to count the non-zero quantum numbers and to estimate, approximately, how many bits per pixel are saved, with respect to 8-bit PCM, by the DCT and quantizer, assuming that no entropy coding is employed. The following simple coding method was assumed for each block of quantum numbers: Send 6 bits for the end-of-block position (there are 64 coefficients per block). For each quantum number out to the end-of-block position, send one zero/not-zero bit, and send 8 bits for each non-zero quantum number. This method gives an upper bound on the number of bits required, and therefore a lower bound on the number saved (8 - number required) by the DCT and quantizer. The number saved by the Q Coder is, then, the difference between the number required by the DCT and quantizer and the number actually transmitted. Since the estimated number of bits required by the DCT and quantizer is an upper bound on the actual number required, the estimated number saved by the Q Coder is also an upper bound.

Figure A1.1 shows, in pie chart form, the bits saved in sequential transmissions of the House with Trees image, where the whole pie represents 8 bits per pixel. It is noteworthy that, for a quantization scale factor of 25 (very good image quality, high bit rate), the Q Coder saves approximately 1.3 bits per pixel, and, at all scale factors, saves roughly half the bits that would be transmitted without entropy coding.

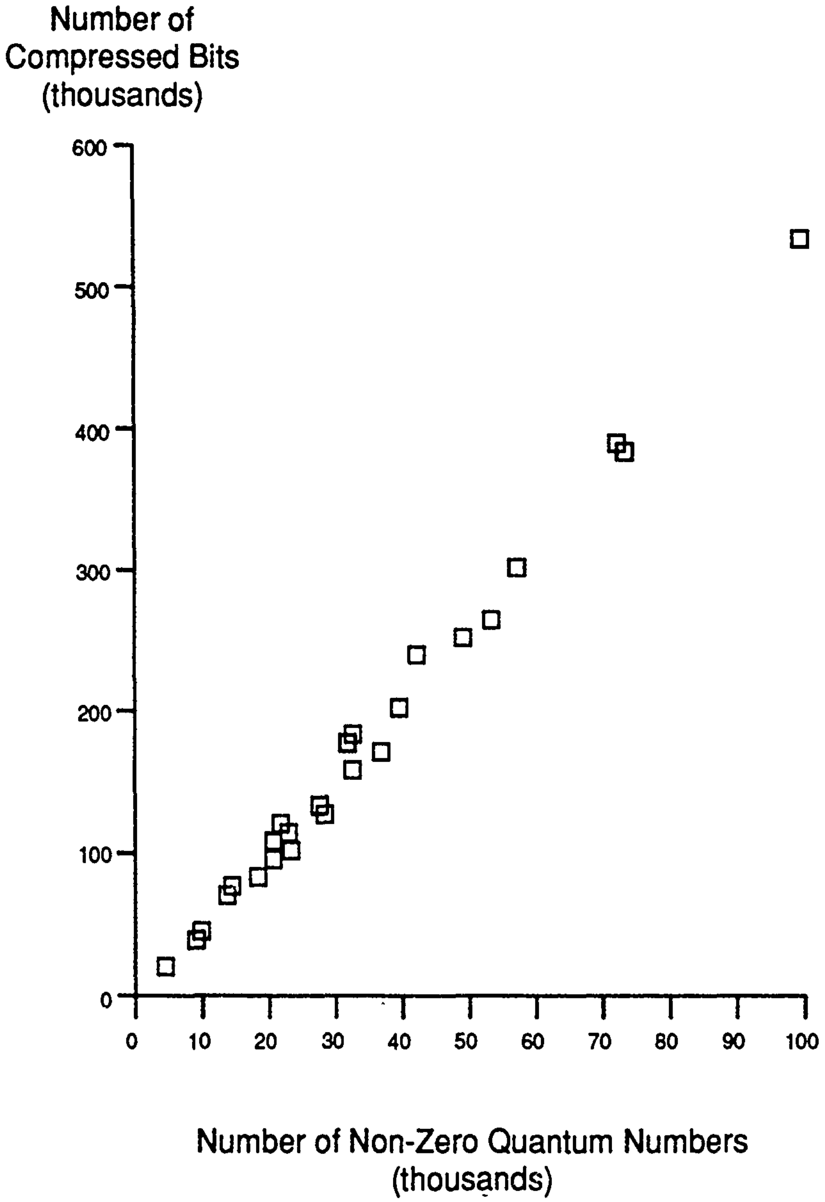


(Whole Pie Represents PCM, i.e. 8 Bits per Pixel)

Image: House with Trees
S = Quantization Scale Factor

Figure A1.1 Data Compression Pie Charts for Sequential Transmissions

Figure A1.2 shows a scatter diagram of number of compressed bits vs. number of non-zero quantum numbers per transmission for all kinds of images and transmission modes except difference "images." To good approximation, each non-zero quantum number requires an average of 5.4 compressed bits.



Note: These data apply to all image types except difference "images."

Figure A1.2 Scatter Diagram of Number of Compressed Bits vs. Number of Non-Zero Quantum Numbers

Difference images require fewer bits per non-zero quantum number, because these quantum numbers usually have smaller magnitudes. The number of bits required were approximately 3.6 in an image-domain "bit slicing" progression and 4.4 in a hierarchical progression.

The value of 3.6 for the "bit slicing" case is surprisingly high, because the magnitudes of the non-zero quantum numbers are always 1, as shown above. Further investigation showed that most of the bits come from the binary decisions for end-of-block, zero/not-zero and sign, with a negligible number from the constant magnitudes of 1 in the non-zero cases. In one test, the compression ratio for the three binary decisions just cited was only 1.15:1, indicating that the results were almost random. Thus, the "overhead" required to transmit a refinement is so large that image-domain "bit slicing" is inefficient. The JPEG bit slicing method gives better compression because it "remembers" the "histories," thus avoiding redundant decisions.

The larger number of bits per non-zero quantum number in a hierarchical refinement is attributed to the fact that the magnitudes of non-zero quantum numbers can be greater than 1. Despite this difference, hierarchical progression gives significantly better compression for a given RMS error than image-domain "bit slicing," although slightly worse than sequential transmission. Examination of the data has shown that fewer non-zero quantum numbers are required to refine an interpolated image in a hierarchical progression than a coarsely-quantized image in image-domain "bit slicing" for similar before-and-after RMS errors. More non-zero quantum numbers are required, however, for an entire hierarchical progression than for a sequential transmission.

The following additional observations may help to explain further why better compression is obtained in hierarchical transmissions than in image-domain "bit slicing:" (1) For a given RMS error in the reconstructed image, one can transmit a 2:1 (in each direction) sub-sampled image with a scale factor of 25 for about the same number of bits as the original image with a scale factor of 200. Thus, the sub-sampled image is accurately encoded with relatively few bits, most of the RMS error coming from interpolation. (2) "Bit slicing" in the image domain requires a great deal of overhead, as observed above, to transmit what amount to 1-bit corrections to the non-zero quantum numbers. Refining an interpolated image requires the same overhead, but the actual quantum number differences may be greater than 1, i.e., the "payload" per non-zero quantum number my be greater than one bit's worth of information. (3) Sub-sampling and interpolation are akin to low-pass filtering. Therefore, in the transform domain, most of the difference between the original and interpolated images is in the higher-frequency coefficients. This tends to emphasize the high-frequency coefficients of the difference "image." Because the "visibility" matrix elements are, in general, larger for high- than for low-frequency coefficients, the high-frequency coefficients are quantized more coarsely than low-frequency ones. This may explain why image refinement in a hierarchical progression requires fewer non-zero quantum numbers than in an image- domain "bit slicing" progression. Nevertheless, these relatively coarsely-quantized high-frequency coefficients are apparently rendered with sufficient accuracy to greatly refine the interpolated image.

REFERENCES

1. STANDARD GRAY SCALE IMAGE USER'S MANUAL, Delta Information Systems, Inc., NCS Contract Number DCA100-83-C-0047

2. Pennebaker and Mitchell, "Simplification to the Algorithm for Arithmetic Coding of the DCT," JPEG, January 7, 1989

3. J. L. Mitchell, W. B. Pennebaker, C. A. Gonzales and C. F. Touchton, "Progress Toward A Color Image Data Compression Standard," a preprint of a paper prepared for presentation at Electronic Imaging '89 West, Pasadena, CA, April 10-13

4. W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, Jr., and R. B. Arps, "An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder," IBM J. Res. Develop., 32, 1988

5. Mitchell and Pennebaker, "Software Implementation of the Q-Coder," IBM J. Res. Develop. 32 1988

6. A. Rosenfeldt and A. Kak, Digital Picture Processing, Second Edition, Volume 1, p. 154, Academic Press, Inc.

7. Ibid, pp. 158-159

8. Ibid, pp. 121 and following

9. Ibid, p. 155

10. Pennebaker, Mitchell and Anderson, "Comparison of Q-Coder vs. Adaptive Huffman," JPEG/No161, July 13, 1988

11. Pennebaker and Mitchell, "EXACT - An Algorithm for Lossless Coding of Continuous Tone Images," JPEG-186, September 10, 1988

12. Dr. Joan L. Mitchell, IBM Thomas Watson Research Center, private communication

13. Pennebaker and Mitchell, "Simplifications to the Algorithm for Arithmetic Coding of the DCT," op. cit.

14. Ibid

15. Pennebaker and Mitchell, private communications

16. Pennebaker and Mitchell, "Simplification to the Algorithm for Arithmetic Coding of the DCT," op. cit.

17. Pennebaker and Mitchell, "EXACT - An Algorithm for Lossless Coding of Continuous Images," op. cit.

18. JPEG 876, 'Proposal for an AC-Prediction Implementation," February, 1989

19."Proposal for an AC-Prediction Implementation," JPEG 876, February, 1989